A SURVEY OF LIMITED NONDETERMINISM IN COMPUTATIONAL COMPLEXITY THEORY

Matthew Asher Levy
University of Kentucky, malevy1@uky.edu

Recommended Citation
Levy, Matthew Asher, "A SURVEY OF LIMITED NONDETERMINISM IN COMPUTATIONAL COMPLEXITY THEORY" (2003). University of Kentucky Master's Theses. 221.
https://uknowledge.uky.edu/gradschool_theses/221
ABSTRACT OF THESIS

A SURVEY OF LIMITED NONDETERMINISM
IN COMPUTATIONAL COMPLEXITY THEORY

Nondeterminism is typically used as an inherent part of the computational models used in computational complexity. However, much work has been done looking at nondeterminism as a separate resource added to deterministic machines. This survey examines several different approaches to limiting the amount of nondeterminism, including Kintala and Fischer’s $\beta$ hierarchy, and Cai and Chen’s guess-and-check model.

KEYWORDS: Computational Complexity, Limited Nondeterminism, Finite Automata

Matthew Asher Levy

December 30, 2002

Copyright © Matthew A. Levy, 2003
A SURVEY OF LIMITED NONDETERMINISM
IN COMPUTATIONAL COMPLEXITY THEORY

By
Matthew Asher Levy

Director: Dr. Judy Goldsmith
Director of Graduate Studies: Dr. Grzegorz W. Wasilkowski
Date: December 30, 2002
RULES FOR THE USE OF THESSES

Unpublished theses submitted for the Master’s degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgements.

Extensive copying or publication of the thesis in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.
A SURVEY OF LIMITED NONDETERMINISM IN COMPUTATIONAL COMPLEXITY THEORY

THESIS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the College of Engineering at the University of Kentucky

By
Matthew Asher Levy
Lexington, Kentucky

Director: Dr. Judy Goldsmith, Professor of Computer Science
Lexington, Kentucky
2003

Copyright ©Matthew A. Levy, 2003
This thesis is dedicated to my loving partner Amy, and to my sons John and Alex.
ACKNOWLEDGMENTS

The following thesis, while primarily an individual work, was the result of a fruitful collaboration, and certainly benefited from the insights and direction of several people. This research was done in collaboration with Judy Goldsmith and Martin Mundhenk, and I thank them both for their cooperation in allowing its use. In addition, my Thesis Chair, Judy Goldsmith, served as a remarkable combination of collaborator and cheerleader. I could not have completed this project without her seemingly infinite patience and support. I would also like to thank Miroslaw Truszczyński and Wiktor Marek, who also served as my advisors for a time, and who exemplify the highest standards of research and scholarship. Next, I wish to thank the complete Thesis Committee: Judy Goldsmith, Mirek Truszczyński and Tony Baxter. Their thoughtful feedback challenged my thinking, substantially improving the final product. I must also give special thanks to my advisor at the University of Southern Maine, Stephen Fenner, who first sparked my interest in computational complexity, and has continued to encourage and support my work.

In addition to the technical and instrumental assistance above, I received equally important assistance from family and friends. My partner, Amy Levy, provided moral and emotional support throughout the thesis process, and listened attentively to my discussions of my work. My boys, John and Alex, have been my inspiration. Without the hope they instill in me, I would never have completed this work.
# TABLE OF CONTENTS

Acknowledgments ........................................ iii

List of Files ........................................... v

Chapter One: Introduction ............................... 1
   A Little Story ........................................ 1
   Description of Research .............................. 1

Chapter Two: Definitions ............................... 4
   Models of Computation ............................... 4
   The Guess-and-Check Model ......................... 4

Chapter Three: The $\beta$ Hierarchy ................. 6
   Polynomial Time ....................................... 6
   Basic properties ...................................... 6
   Complete problems ................................... 7
   Finding a “Small” Forcing Set ....................... 9
   Other Structural Properties ......................... 11
   Below and Above Polynomial Time ................... 12
   Real-Time Computations .............................. 13
   Relations between the $\beta$-Hierarchies .......... 14

Chapter Four: Quasilinear Time ....................... 15
   Basic Properties ..................................... 15
   Complete Sets ........................................ 16

Chapter Five: Limited Nondeterminism in Other Settings ........................................ 18
   Maximization Problems ............................... 18
   Finite Automata ...................................... 19

References ............................................. 21

Vita ................................................... 24
List of Files

<table>
<thead>
<tr>
<th>FILE</th>
<th>TYPE</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>levythes.pdf</td>
<td>Acrobat PDF</td>
<td>292K</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

A Little Story

Amy is a systems analyst working for a university electrical engineering department. The EE people want to be able to give her a circuit and find out if a polylogarithmic number of ones in the input can force the circuit to output a one. Amy has a BS in computer science, so she knows that $\text{Csat}$ (circuit satisfiability) is NP-complete. But the conditions seem restrictive enough that she decides to give it a try. First, she writes a program to see if a logarithmic number of ones will force a circuit to one. This program works well, and it doesn’t take too long, even on reasonably big circuits. Her superiors are impressed, and so she decides to tackle $\log^2 n$ ones.

Amy tries the same approach, and her program works for small circuits. But, when she tries larger circuits, the program runs for ages! Her bosses are getting impatient, but no algorithm she can think of can deal with the large circuits effectively. She remembers her CS training, and decides that this must be NP-complete. Armed with a compendium of reductions, Amy sits down to prove that her problem is NP-complete....

After running through several reams of paper, Amy’s enthusiasm wanes. None of her techniques help. The problem is in NP, although it only takes $\log^2 n$ guesses, but she can’t reduce any NP-complete problem to it. What’s she going to do?

This story is essentially the same one we tell about NP-completeness. The difference is, in this case, the problem actually has a subexponential algorithm, and doesn’t seem to be NP-complete. Can we convince Amy’s bosses that this problem is inherently very hard?

The key is in the nondeterministic solution. The algorithm never needs more than $\log^2 n$ nondeterministic bits. If Amy were more familiar with limited nondeterminism, she would know the answer.

As it turns out, this story has a happy ending. Amy has lunch with her theorist friend Alex, and discusses the problem. Alex does a little research, and finds that Amy’s problem is complete for a class called $\beta_2 \text{P}$. Every problem in $\beta_k \text{P}$ needs at most $\log^k n$ nondeterministic bits and polynomial time. Since $\beta_1 \text{P} = \text{P}$, Amy was able to solve the problem for the logarithmic case. Alex writes up an explanation for Amy’s bosses, who chalk it up to experience and start looking for approximations.

---

1Most of this work first appeared in [19].
Description of Research

Amy’s situation goes right to the heart of some of the central questions in complexity theory: What is the power of nondeterminism in bounded-resource computation? Can nondeterminism decrease the time complexity of some problems? If so, how much nondeterminism is needed to make the difference?

There are other problems than Amy’s in NP that do not use the full nondeterministic power of NP. For instance, Pratt’s algorithm \cite{34} solves primality with only $O(n^2)$ nondeterministic bits; computing the Vapnik-Chervonenkis (V-C) dimension requires only $O(\log^2 n)$ nondeterministic bits \cite{33}. These problems come from a wide variety of fields within computer science: cryptography and computational number theory (Pratt’s algorithm, for instance); learning theory (the V-C dimension); formal language theory (Unary Generators \cite{9}) and computational geometry \cite{6}, to name a few.

The latter results are concerned with limited nondeterminism in P and in NP. Other nondeterministic classes can also be restricted in natural ways. In this survey, we will also examine the power of limited nondeterminism in real-time, within nondeterministic NC circuits, and even in NE. In the work surveyed, the bounds on nondeterminism are given \textit{a priori}. There are also classes which implicitly use limited nondeterminism and alternation. For example, the classes LOGNP and LOGSNP, defined by Papadimitriou and Yannakakis \cite{33}, are both contained in $\beta_2 P$. We look at their work in Chapter 5.

There are also hierarchies defined by both limited nondeterminism and some form of alternation, such as the W-hierarchy \cite{15}, a hierarchy within NP of parameterized problems \cite{13}, and the Sharply Bounded Hierarchy, an alternation-based hierarchy built on quasilinear time and $O(\log n)$-bit existential and universal quantifiers \cite{5, 9}.

Extensive studies are done on limited nondeterminism in the context of finite automata and pushdown automata. (See \cite{20, 21, 27, 30, 29}, and \cite{38} for finite automata, and \cite{35, 36}, and \cite{40} for pushdown automata.) In this context, the limits on nondeterminism are more likely to be a function of the size of the automaton than of the length of the input for a computation. We explore some of these results in Chapter 5.

Certainly, nondeterminism is a powerful tool, and has been applied in many different settings. This survey does not address the relationship of restricting nondeterminism to restricting other resources which are strongly connected to nondeterminism (e.g., randomness), nor approximability issues \cite{11}. Our goal is to provide an introduction to some of the work in a potentially rich, but still largely unmined, area of research. We also present some new results in the section 3.4 on page 9 in Chapter 3, describing the complexity of the $\beta_k$-SAT \cite{17, 39} problem (see Definition 3.6 on page 8) in more detail.
Chapter 2
Definitions

Turing machines are considered here as multitape Turing machines; nondeterministic steps of Turing machines are binary choices. Kintala and Fischer [26] considered time bounded Turing machines which make a bounded number of nondeterministic steps. In a sense, nondeterminism is seen as an additional resource for deterministic computations, and the deterministic and nondeterministic complexity of a computation is measured separately.

Models of Computation

Definition 2.1 (cf. [26]) Let \( \mathcal{F} \) be a class of functions and \( \mathcal{C} = \bigcup_{f \in \mathcal{F}} \text{DTIME}(f) \) be the class of sets accepted by deterministic \( \mathcal{F} \)-time bounded Turing machines. For a total function \( g : \mathcal{N}_0 \to \mathcal{N}_0 \), \( g(n) \cdot \mathcal{C} \) denotes the class of sets accepted by nondeterministic \( \mathcal{F} \)-time bounded Turing machines making at most \( g(n) \) nondeterministic steps on every input of length \( n \).

This model of limited nondeterminism restricts unlimited nondeterministic time-bounded computations. In order to add nondeterminism to complexity classes independent of the computational model for the classes, Cai and Chen [12] defined the so-called Guess-and-Check model.

The Guess-and-Check Model

In the Guess-and-Check model, the nondeterministically chosen bits are appended to the input. If one of the choices can be verified as a witness, the original input is accepted. This model can also be used to describe e.g. nondeterministic circuit classes.

Definition 2.2 [12] Let \( s(n) \) be a function and let \( \mathcal{C} \) be a complexity class. Then \( s\text{GC}(b(n), \mathcal{C}) \) is defined to be the class of all sets \( A \), for which there is a set \( B \in \mathcal{C} \) such that for every \( x \), \( x \in A \iff \exists z, |z| \leq b(|x|) \) such that \( \langle x, z \rangle \in B \).

Some classes have similar characterizations in both models. For instance, adding polynomial nondeterminism to P yields NP, i.e. \( \bigcup_c n^c \cdot P = \bigcup_c s\text{GC}(n^c, P) = \text{NP} \). On the other hand it is not clear if both these models can capture the same classes, since in the GC model the witness can be used to increase the computation time. For example, for \( E = \bigcup_c \text{DTIME}(2^{n^c}) \), the class \( \bigcup_c s\text{GC}(n^c, E) \) contains \( \text{EXP} = \bigcup_c \text{DTIME}(2^{n^c}) \), whereas it is unknown whether

\[^1\mathcal{N} \text{ denotes the set of positive natural numbers } \{1, 2, \ldots\}, \mathcal{N}_0 \text{ denotes } \{0\} \cup \mathcal{N}.\]
EXP is contained in $\bigcup_n^c n^c$-E, a subclass of NE. Similar observations can be made for the space bounded case.\footnote{Definition 2.1 can be stated for space bounded computations in a natural way.} There, the witness can be used as additional computation space, yielding $\bigcup_n^c n^c$-L = NL $\subseteq$ NP = $\bigcup_n^c sGC(n^c, L)$. 

Copyright © Matthew A. Levy, 2003
Chapter 3

The $\beta$ Hierarchy

Kintala and Fischer [25, 26] defined classes of sets accepted by polynomial-time bounded Turing machines that make a polylogarithmic number of nondeterministic steps. These seminal papers motivated consideration of variants of these classes. We present results on real-time, polynomial-time, and exponential-time computations with limited nondeterminism.

Polynomial Time

Kintala and Fischer [26] defined the classes $g(n)$-P as in Definition 2.1.

Using exhaustive search, every polynomial-time computation with a logarithmic number of nondeterministic steps can be simulated by a deterministic polynomial-time computation. I.e. in this notation $P = (c \log n)$-P for every $c \in \mathbb{N}_0$. Furthermore, since no polynomial-time Turing machine can make more than a polynomial number of nondeterministic steps, NP is equal to the union over all polynomials $f$ of the classes $f(n)$-P.

Using standard padding arguments, one can show that there exist NP-complete sets in $n^\varepsilon$-P for every $\varepsilon > 0$. The original goal in [26] was to define candidates for sets which are neither in P nor NP-complete, and thus the focus was on polynomial-time machines that use a subpolynomial and superlogarithmic amount of nondeterminism. This led to the definition of the $\beta$-hierarchy over P.

**Definition 3.1** (cf. [26]) For every $k \in \mathcal{N}$, $\beta_k P$ is defined as the class $\bigcup \epsilon (c \log^k n)$-P. The $\beta P$-hierarchy is $\beta P = \bigcup_{k \in \mathcal{N}} \beta_k P$.

As mentioned above, $P = \beta_1 P$. The $\beta P$-hierarchy consists of $\beta_1 P \subseteq \beta_2 P \subseteq \cdots \subseteq \bigcup_{k \in \mathcal{N}} \beta_k P$ and lies between P and NP (cf. [26]). Note that $\beta_k P = \bigcup_c sGC(c \log^k n, P)$ (see [12]).

**Basic properties**

If $P = NP$, then $\beta P = \beta_1 P = \beta_2 P = co\beta_j P$ for all $i, j \in \mathcal{N}$. If the $\beta P$-hierarchy consists of at least two levels (i.e. if $\beta_i P \subseteq \beta P$) or if at least one of its levels is not closed under complementation (i.e. $\beta_i P \neq co\beta_i P$ for some $i$), then $P \neq NP$. It is not known whether any “minor” collapse, such as $\beta_k P = co\beta_k P$ (for $k > 1$) or $\beta P = \beta_k P$, would have any consequences on the collapse of other classes in the hierarchy, or even on other complexity classes.
However, such consequences would not be provable with known relativizing techniques. Beigel and Goldsmith [4] showed that “everything goes;” the classes of the $\beta$P-hierarchy can be collapsed, separated, or closed under complementation in each way which is consistent with its general inclusion structure given above — relative to oracles. “Consistent” means that the unrelativized inclusion structure of the $\beta$P-hierarchy and its complement must not be injured.

**Theorem 3.2** [4] Imagine any consistent inclusion structure of the $\beta$P-hierarchy. There exists an oracle relative to which the $\beta$P-hierarchy has this structure.

**Complete problems**

Properties which distinguish the $\beta$P-hierarchy and NP are known only in relativized worlds. The next two theorems show similarities between NP and classes in the $\beta$P-hierarchy.

**Theorem 3.3** [2] $\beta_k P$ is closed under polynomial-time disjunctive reducibility, for every $k \in \mathbb{N}$.

This implies that every $\beta_k P$ is also closed under polynomial-time and logspace many-one reductions (denoted $\leq^P_m$ and $\leq^L_m$). It is open whether $\beta_k P$ is closed under polynomial-time conjunctive reducibility. In [2] this question is answered negatively in a relativized world.

**Theorem 3.4** (cf. [2, 12, 17, 26, 39]) For every $k \in \mathbb{N}$, $\beta_k P$ has $\leq^L_m$-complete sets.

In [26] “generic” complete sets were shown to be complete for $\beta_k P$. Those sets are

$$\beta_k-C = \left\{ (M, x, 1^t) \mid M(x) \text{ accepts in } \leq t \text{ steps using } \leq \log^k |x| \text{ nondeterministic steps} \right\}.$$

In [2] it is proved that the following variants of the circuit value problem are complete for the $\beta_k P$ classes:

$$\beta_k-C_{\text{VP}} = \left\{ (x, C) \mid C \text{ is a circuit with } |x| + \lceil \log^k |x| \rceil \text{ inputs which outputs 1 on some input } xz \right\}.$$

These are generalizations of P-complete sets, leading to the conjecture that no restriction of an NP-complete set would be $\beta_k P$-complete [2]. Cai and Chen [12], Farr [17] and Szelepćsényi [39] disproved that conjecture.
Cai and Chen [12] considered restrictions of the weighted circuit satisfiability problem. They proved, for every $k \in \mathbb{N}$, the $\beta_k P$-completeness of

$$\text{BWCS}_k = \left\{ C \mid C \text{ is a circuit with } m \text{ inputs which outputs } 1 \text{ on some input containing at most } \log^{k-1} m \text{ many 1s} \right\}.$$ 

As a more involved example we give the definition of a variant of the NP-complete set SAT of satisfiable Boolean formulae $\phi$ in conjunctive normal form, i.e. $\phi$ is a set of clauses of literals.

For a formula $\phi$ and a partial assignment $\mathcal{A}$ of literals from $\phi$, let $\phi^\mathcal{A}$ denote the formula obtained from $\phi$ by deleting all clauses which contain a literal in $\mathcal{A}$ and all literals which complements are in $\mathcal{A}$. If one of the clauses becomes empty, then $\mathcal{A}$ does not satisfy $\phi$. Every unit clause $\{l\}$ in $\phi$ forces any satisfying assignment including $\mathcal{A}$ to contain $l$. If iterating this process leads to an empty formula, we say that $\mathcal{A}$ forces a satisfying assignment for $\phi$, denoted $\mathcal{A} \vdash \phi$.

**Definition 3.5** Let $\phi$ be a boolean formula. Then $v(\phi)$ is defined as the set of all variables appearing in $\phi$.

**Definition 3.6** Define $\beta_k$-SAT as follows:

$$\beta_k \text{-SAT} = \left\{ \langle \phi, V \rangle \mid \phi \text{ a formula, } V \subseteq v(\phi), |V| \leq \lceil \log^k |\phi| \rceil, \text{ such that } \right\}$$

there exists an assignment $\mathcal{A}$ for $V$ where $\mathcal{A} \vdash \phi$.

Note that $\beta_1$-SAT is $p$-isomorphic to the P-complete complement of the unit resolution problem.

**Theorem 3.7** [17, 39] For every $k \in \mathbb{N}$, $\beta_k$-SAT is $\leq^L_m$-complete for $\beta_k P$.

The above algorithm can be described as “greedy” [39], or by saying that an assignment of Boolean values to the variables is “forced” [17]. The proofs of Theorem 3.7 can be extended to other P-complete problems solvable having “greedy” or “forcing” algorithms. The $\beta_k$P-completeness of variants of 3SAT, vertex cover, clique, Hamiltonian circuit, and 3-dimensional matching are proved in [39], and of restrictions of 3-colourability and Hamiltonian circuit in [17].
Finding a “Small” Forcing Set

The algorithm described above for $\beta_k$-Sat is an integral part of nonmonotonic reasoning, known as the well-founded semantics (see for example [31]). The smodels system implemented by Niemelä and Simons ([31]) achieves excellent performance in many cases by finding small (i.e., log $n$ size) forcing sets.

While $\beta_k$-Sat $\in \beta_k$, the problem definition includes a candidate forcing set in the input. This does not include the potential cost of finding such a forcing set. In the course of this work, we examined this complexity independently. First, we define the problem Log$_k$-Sat, which takes a formula $\phi$ and identifies whether or not a log$^k n$ size forcing set exists:

**Definition 3.8** Define the set Log$_k$-Sat as follows:

$$\text{Log}_k\text{-Sat} = \{\langle \phi, V \rangle \mid \exists V' \supseteq V, \langle \phi, V' \rangle \in \beta_k\text{-Sat} \}$$

Note that we have included a (possibly empty) set of variables that must appear in the desired forcing set. Observe that we can determine if any “small” forcing set exists for a formula $\phi$ by computing Log$_k$-Sat($\langle \phi, \emptyset \rangle$).

Using a padding argument, we can show that Log$_k$-Sat is hard for $\beta_k$-Sat.

**Theorem 3.9** $\beta_k$-Sat $\leq^p_m$ Log$_k$-Sat.

**Proof** Let $\langle \phi, V \rangle$ be an input for $\beta_k$-Sat. Define the reduction

$$f(\langle \phi, V \rangle) = \begin{cases} \langle \phi, V \rangle & \text{if } |V| = \lceil \log^k n \rceil \\ \langle \phi_{\text{ext}}, V_{\text{ext}} \rangle & \text{if } |V| < \lceil \log^k n \rceil \\ \bot & \text{otherwise} \end{cases}$$

where $\phi_{\text{ext}}$ and $V_{\text{ext}}$ are constructed as follows. In step 0, initialize $\phi_0 = \phi$, $V_0 = V$ and $\gamma_0 = \lceil \log^k |\phi| \rceil - |V|$. We also define the formula $\psi(a,b) = x_{a+2b} \lor x_{a+2b+1}$ for all $0 \leq a, b$. In step $i > 0$:

1. Let $\gamma_i = \lfloor \log^k |\phi_{i-1}| \rfloor - |V_{i-1}|$. Let $\delta_i = \gamma_i - \gamma_{i-1}$ and $v_i = |v(\phi_{i-1})| + 1$. ($\delta_i$ is the number of new clauses we’re adding, and $v_i$ is the starting index of the first new variable we add.)

2. Set

$$\phi_i = \phi_{i-1} \land \left( \bigwedge_{j=0}^{\delta_i} \psi(v_i, j) \right)$$

and

$$V_i = V_{i-1} \cup \left( \bigcup_{j=1}^{\delta_i} x_{v_i+2j} \right)$$

9
3. If \( V_i = V_{i-1} \), set \( \phi_{ext} = \phi_i, V_{ext} = V_i \) and \( \gamma = |V_i| - |V| \), and halt.

This algorithm must terminate, because \( \log^k n \in o(n) \).

The reason we do this padding is that it is possible that if \( |V| < \lfloor \log^k |\phi| \rfloor \) there might be no assignment of \( |V| \) that forces \( \phi \), but some superset of \( V \) might force \( \phi \) without exceeding our \( \lfloor \log^k |\phi| \rfloor \) bound. If we simply used the identity function as our reduction, this case would cause \( f(\langle \phi, V \rangle) \in \text{Log}_k\text{-Sat} \) even though \( \langle \phi, V \rangle \notin \beta_k\text{-Sat} \). Observe that we have ensured that for all \( f(\langle \phi, V \rangle) = \langle f_1(\phi), f_2(V) \rangle \), it is the case that \( |f_2(V)| = \log^k |f_1(\phi)| \).

We can also characterize \( \phi_{ext} \) and \( V_{ext} \) as:

\[
\phi_{ext} = \phi \land \phi' \\
\phi' = (x'_1 \lor x'_2) \land \cdots \land (x'_{2\gamma-1} \lor x'_{2\gamma})
\]

and

\[
V_{ext} = V \cup V' \\
V' = \{x'_2, x'_4, \ldots, x'_{2\gamma}\}
\]

That is, we’ve added \( 2\gamma \) new variables, and added additional clauses of the form \( (x'_j \lor x'_{j+1}) \) using those new variables. These additions have an important property.

**Lemma 3.10** Any assignment to the variables in \( V' \) forces a satisfying assignment of \( \phi' \).

**Proof** Let \( \phi' = C_1 \land C_2 \land \cdots \land C_{\gamma} \), where \( C_i = x'_{2i-1} \lor x'_{2i} \). Let \( A_{V'} \) be a partial assignment of \( \nu(\phi_{ext}) \) to all of the variables in \( V' \). For any clause \( C_i, A_{V'}(x'_{2i}) \) is defined and \( A_{V'}(x'_{2i-1}) \) is not (initially) defined. If \( A_{V'}(x'_{2i}) = 1 \), \( C_i \) is satisfied, so suppose not. Then, during the unit propagation phase, we will remove \( x'_{2i} \) from \( C_i \), leaving the singleton clause \( \{x'_{2i-1}\} \), which forces \( A_{V'}(x'_{2i-1}) = 1 \). Hence, in either case, \( C_i \) is satisfied. Since neither \( x'_{2i} \) nor \( x'_{2i-1} \) appears in any other clause, this does not affect the forcing of any other variable. Since \( i \) was arbitrary, \( A_{V'} \) must force every clause to be satisfied, and hence \( A_{V'} \vdash \phi' \). We started with an arbitrary assignment, so it must be that for any assignment \( A, A \vdash \phi' \). \( \square \)

The function \( f \) is computable in polynomial-time. We must now show that \( f \) is a reduction.

Let \( \langle \phi, V \rangle \in \beta_k\text{-Sat} \). If \( f(\langle \phi, V \rangle) = \langle \phi, V \rangle, \text{Log}_k\text{-Sat}(\langle \phi, V \rangle) = 1 \), since \( V \subseteq V \). Suppose \( f(\langle \phi, V \rangle) = \langle \phi_{ext}, V_{ext} \rangle \). If we consider \( \phi \) and \( \phi' \) separately, we see that \( V \) must force a satisfying assignment of the clauses in \( \phi \), since we assumed that \( \langle \phi, V \rangle \in \beta_k\text{-Sat} \). And, by Lemma 3.10, \( V' \) forces a satisfying assignment of the clauses in \( \phi' \). So, \( f(\langle \phi, V \rangle) \in \text{Log}_k\text{-Sat} \).

Now, let \( \langle \phi, V \rangle \notin \beta_k\text{-Sat} \). If \( f(\langle \phi, V \rangle) = \langle \phi, V \rangle, \text{Log}_k\text{-Sat}(\langle \phi, V \rangle) = 0 \), since \( V \) is the only superset of \( V \) of length \( \log^k n \). Suppose \( f(\langle \phi, V \rangle) = \langle \phi_{ext}, V_{ext} \rangle \). By construction,
$V_{\text{ext}}$ is the only superset of $V_{\text{ext}}$ of length $\lceil \log^k n \rceil$, so it is the only possible witness for membership. Although the clauses in $\phi'$ are satisfied by any assignment, there is no assignment that can satisfy $\phi$, by assumption. Hence, $\phi_{\text{ext}}$ cannot be satisfied. Therefore, $\text{Log}_k\text{-SAT}(f((\phi_{\text{ext}}, V_{\text{ext}}))) = 0$.

In order to show $\beta_k$-completeness, we would have to show $\text{Log}_k\text{-SAT} \in \beta_k$, but this seems unlikely to be the case. (We explain the rationale for this statement below.) However, we can show that $\text{Log}_k\text{-SAT} \in \beta_{k+1}$.

**Theorem 3.11** $\text{Log}_k\text{-SAT} \in \beta_{k+1}$.

**Proof** First, guess a subset $V' \subseteq X, V \subseteq V'$ by writing down the indices of the variables in $V'$. Each index takes $\log n$ bits to write down, and $|V'| = \log^k n$, so we are guessing $(\log^k n)(\log n) = \log^{k+1} n$ bits. Now, we guess an assignment for the variables in $V'$, which takes $\log^k n$ bits. We use that assignment to verify that $V'$ forces a satisfying assignment of $\phi$, which takes polynomial time. \qed

As mentioned above, it is natural to wonder if $\text{Log}_k\text{-SAT}$ is complete, either for $\beta_k$ or $\beta_{k+1}$. These both seem unlikely. It is doubtful that $\text{Log}_k\text{-SAT} \in \beta_k$, since there is no obvious way to represent $\log^k n$ elements using fewer than $\log^{k+1} n$ bits. $\text{Log}_k\text{-SAT}$ also seems unlikely to be hard for $\beta_{k+1}$. Consider $\beta_{k+1}\text{-SAT}$, which is $\beta_{k+1}$-complete. There are $n^{\log^k n}$ possible assignments for any set of $\log^{k+1} n$ variables. The number of $\log^k n$ size subsets (which we use in computing $\text{Log}_k\text{-SAT}$) is $\binom{n}{\log^k n}$. We can use Stirling’s approximation to see that $\binom{n}{\log^k n} \in o(n^{\log^k n})$. That is, the number of possible assignments grows faster than the number of subsets, and we would have to be able to compress those assignments, in polynomial-time. This seems extremely difficult, if not impossible, barring a collapse of $\beta P$.

**Other Structural Properties**

There are more structural similarities between classes in $\beta P$ and NP. Several equivalent characterizations of the $P = NP$ question can be translated to the $P = \beta_k P$ question. In [3] it is shown that $P = NP$ if and only if an NP-complete set reduces to some sparse set via a certain type of reducibility. A set is sparse if it has at most polynomially many elements of each length. Informally, this result gives a lower bound on the information content of NP-complete sets. The same lower bound can be proven for sets in $\beta P$.

**Theorem 3.12** [3] For $k \in \mathcal{N}$, $P = \beta_k P$ iff a $\leq^L_{m}$-complete set for $\beta_k$ polynomial-time reduces via a bounded truth-table reducibility to a set which polynomial-time conjunctively reduces to a sparse set.
For every $\varepsilon > 0$, NP is the closure of NTIME($n^\varepsilon$) under logspace reductions, but every $\beta_k P$ is closed under these reductions. This yields

**Theorem 3.13** [37] For any $k \in \mathcal{N}$, $\beta_k P = NP$ iff NTIME($n^\varepsilon$) $\subseteq \beta_k P$ for some $\varepsilon > 0$.

There are some differences between the $\beta_k P$ and the $N^P_m$ classes defined in Chapter 4. For instance, the $N^P_m$ classes are closed under quasilinear-time reductions, but not under polynomial-time reductions. Furthermore, Theorems 3.13 and 4.5 show that these hierarchies relativize in dramatically different ways.

One can also characterize $\beta P$ in terms of oracle computations. The following characterization uses nondeterministic polylogarithmic time bounded oracle machines as defined in [24]. Since polylogarithmic time does not suffice to copy the input on the oracle tape, it is assumed that the input is written on the oracle tape for free.

**Theorem 3.14** [2] $\beta P = NTL^P$.

Another characterization of $\beta P$ is obtained using robust oracle machines. In [28] it is shown that NP is the class of sets accepted by one-sided robust polynomial-time bounded oracle machines using an NP oracle, formally $P^{NP\text{-help}}_1 = NP$. Bounding the number of queries to the oracle yields a characterization of $\beta P$.

**Theorem 3.15** [2] $P^{NP\text{-help}}_1 \log^k n \subseteq P^{\beta_k P\text{-help}}_1 \log^k n = \beta_k P$.

**Below and Above Polynomial Time**

Hemaspaandra and Jha [23] considered limited nondeterminism in linear-exponential time computations and defined the $\beta$-*hierarchy over $E$.*

**Definition 3.16** [23] For every $k \in \mathcal{N}$, $\beta_k E$ is defined as $\bigcup_{e} (cn^k)$-$E$. The $\beta E$-hierarchy is $\beta E = \bigcup_{k \in \mathcal{N}} \beta_k E$.

Observe that $\beta_1 E = E$. Like $\beta P$, it is unknown whether $\beta E$ is a proper hierarchy. In [23], it is shown that the $\beta E$-hierarchy, like the $\beta P$-hierarchy, doesn’t have the downward separation property — relative to some oracles.

**Theorem 3.17** [23] For every $k \in \mathcal{N}$, there exists an oracle relative to which

$$E = \beta_1 E = \beta_k E \subset \beta_{k+1} E \subset \beta_{k+2} E \subset \cdots$$
Wolf [41] defined nondeterministic circuits, which are allowed to have nondeterministic input gates. Let $\text{NC}^i$ denote the class of sets decided by logspace uniform circuits of polynomial size and depth $\log^i n$, and $\text{NC} = \bigcup_i \text{NC}^i$.

**Definition 3.18** [41] For every $k \in \mathbb{N}$, $\beta_k \text{NC} = \bigcup_c s\text{GC}(c \log^k n, \text{NC})$. The $\beta\text{NC}$-hierarchy is $\beta\text{NC} = \bigcup_{k \in \mathbb{N}} \beta_k \text{NC}$.

In [41] it is shown that quasigroup isomorphism, Latin square isomorphism and Latin square graph isomorphism problems are contained in $\beta_2 \text{NC}^2$.

Similarly to the nondeterministic generalizations of P, all nondeterministic NC classes are between NC and NP, and have complete problems.

**Theorem 3.19** 1. [41] $\beta_1 \text{NC} = \text{NC}$, and $\text{NP} = \bigcup_{k \in \mathbb{N}} s\text{GC}(n^k, \text{NC})$.

2. [12] $\beta_k \text{NC}^i$ contains $\leq_m$-complete sets for for all $i \in \mathbb{N}$, $k \in \mathbb{N}_0$.

A similar result as in Theorem 3.13 for time-bounded classes can be shown for space-bounded classes.

**Theorem 3.20** [41] For any $k \in \mathbb{N}$, if $\text{SPACE}(n^\varepsilon) \subseteq s\text{GC}(n^k, \text{NC})$ for some $\varepsilon > 0$, then $\text{PSPACE} = \text{NP}$.

In [5] a similar assumption for time-bounded computations yields a stronger consequence: if $\bigcup_r \text{DTIME}(n^k \log^r n) \subseteq \text{SBH}$ for some $k > 1$, then $\text{P} = \text{PSPACE}$, where SBH is the Sharply Bounded Hierarchy defined in [5].

**Real-Time Computations**

A real-time machine (cf. [8]) is a (multi-tape) Turing machine which reads a new input symbol at every step. Let $R$ (resp. $\text{NR}$) denote the class of all sets accepted by deterministic (resp. nondeterministic) real-time machines. Book and Greibach [8] showed that $\text{NR}$ is the class of sets accepted by nondeterministic linear-time Turing machines. $g(n)$-$R$ is defined as in Definition 2.1.

Observe that $R = c$-$R$ for every constant $c \in \mathbb{N}_0$ and $\text{NR} = n$-$R$. Unlike the nondeterministic generalizations of P and E, exhaustive search cannot be used to show whether $R$ equals $(\log n)$-$R$. Hartmanis and Stearns [22] showed that $R \subset \text{NR}$. Fischer and Kintala [18] refined the result by showing that there is an infinite hierarchy of classes between $R$ and $(\log n)$-$R$. 
Theorem 3.21 \[18\] Let \( g, h \) be real-time constructible functions, \( g(n) \leq \log n \), and \( h(n) = o(g(n)) \). Then \( h(n)-R \subset g(n)-R \).

This yields the following proper hierarchy of classes between \( R \) and \((\log n)-R\), where \( \log^{(k)} n = \log \cdots \log n \): \( R \subset \cdots \subset (\log^{(k+1)} n)-R \subset (\log^{(k)} n)-R \subset \cdots \subset (\log n)-R \).

Remember that this distinguishes real-time computations from the polynomial-time or linear exponential-time computations considered above, since \( P = (\log n)-P \) and \( E = n-E \). But it is unknown whether the hierarchy of classes obtained by real-time computations with polylogarithmically many nondeterministic steps is also proper.

Definition 3.22 \( \beta_k R \) denotes the class \( \bigcup_c (c \log^k n)-R \). The \( \beta R \)-hierarchy is \( \beta R = \bigcup_{k \in \mathbb{N}} \beta_k R \).

The \( \beta R \)-hierarchy has the inclusion structure \( R \subset \beta_1 R \subset \beta_2 R \subset \cdots \subset \beta R \subset NR \).

Relations between the \( \beta \)-Hierarchies

The structure of the \( \beta R \)-hierarchy is closely related to the structure of the \( \beta \)-hierarchies over \( P \) and \( E \). Using padding arguments, one can relate collapses of \( \beta R \) to collapses of \( \beta E \).

Theorem 3.23 Let \( k \in \mathbb{N} \).

1. \[18\] \( \beta_k R = \beta_{k+1} R \) implies \( \beta_k P = \beta_{k+1} P \), and \( \beta_k R = \text{co} \beta_k R \) implies \( \beta_k P = \text{co} \beta_k P \).
2. \[23\] \( \beta_k P = \beta_{k+1} P \) implies \( \beta_k E = \beta_{k+1} E \), and \( \beta_k P = \text{co} \beta_k P \) implies \( \beta_k E = \text{co} \beta_k E \).

Book \[7\] showed that the collapse of \( E \) and \( NE \) is strongly related to the structure of \( NP \), namely \( E = NE \) iff \( NP \) – \( P \) contains a tally set \( T \subseteq 0^* \). Hemaspaandra and Jha proved this result also for classes of limited nondeterminism between \( E \) and \( NE \). The proof takes advantage of the fact that for every \( A \in E \) (resp. \( A \in \beta_k E, A \in NE \)), the set \( A\#^{2^n} = \{ x\#^{2^{|x|}} \mid x \in A \} \) is in \( P \) (resp. in \( \beta_k P, NP \)), and \( A\#^{2^n} \) is \( p \)-isomorphic to a tally set.

Theorem 3.24 Let \( k \in \mathbb{N} \).

1. \[18\] \( \beta_{k+1} P = \beta_k P \) iff for every \( A \in \beta_{k+1} R \) exists a constant \( c \in \mathbb{N} \) such that \( A\#^{c^c} \in \beta_k R \).
2. \[23\] \( \beta_{k+1} E = \beta_k E \) iff there is no tally set in \( \beta_{k+1} P – \beta_k P \).

Copyright © Matthew A. Levy, 2003
Chapter 4
Quasilinear Time

Many basic algorithmic problems, e.g. multiplication and sorting, can be performed in quasilinear time, i.e. in time $O(n \log^k n)$ for some constant $k$. The log$^k n$ factor in the time bound makes the class of quasilinear time decidable sets robust. For example, it is independent of the number of tapes of a multitape Turing machine, and is closed under quasilinear time reductions, denoted $\leq^{QL}_m$. There are interesting open problems concerning quasilinear time bounds. It is unsolved whether a circuit can be topologically ordered in quasilinear time on a Turing machine, whereas it is known that quasilinear time suffices for this task on a random access machine.

Definition 4.1 [9] Let $l \in \mathbb{N}$ and $m \in \mathbb{N}_0$. Then $P_l$ denotes the class $\bigcup_r \text{DTIME}(n^l \log^r n)$, $NP_l$ is $\bigcup_r \text{NTIME}(n^l \log^r n)$, and $N^mP_l$ is $(m \log n)\cdot P_l$.

Using exhaustive search to simulate nondeterministic computations deterministically yields:

Observation 4.2 [9] For any $m$ and $l$, $N^mP_l$ is contained in $P_{m+l}$, and $\bigcup_{m,l \in \mathbb{N}} N^mP_l = P$.

Basic Properties

These classes fall into the Guess-and-Check model: $N^mP_l = sGC(m \log n, P_h)$ (cf. [12]). This hierarchy exhibits the downward separation property w.r.t. both time and nondeterminism.

Theorem 4.3 [9] For all $m, l \in \mathbb{N}_0$,

1. if $N^lP_l \subseteq P_l$, then $N^kP_l \subseteq P_l$ for all $k \geq l$, and

2. if $N^{m+1}P_l \subseteq N^mP_l$, then $N^kP_l \subseteq N^mP_l$ for all $k \geq m$.

In fact, this theorem relativizes. The hierarchy also exhibits a kind of upward separation:

Theorem 4.4 [9] Suppose that for all $l$, there is an $m$ such that $N^mP_1$ is not contained in $P_l$. Then $P \neq NP$.

Unlike the $\beta P$-hierarchy, there are only countably many relativized configurations of this hierarchy possible.
Theorem 4.5 [9] There are oracles relative to which:

1. $N^mP_k = P_k$ for all $m$ and $k$, and yet $\text{NP} \not\subseteq P$.

2. $P_m \neq N^3P_m$ and $N^3P_m = \text{NP}_m$ for all $m$.

3. $\text{NP}_j \neq N^4P_l$ for all distinct pairs $(i, j)$ and $(k, l)$ and also $P \neq \text{NP}$.

Complete Sets

Theorem 4.6 [9] $N^mP_l$ contains $\leq^{QL}_m$-complete sets for all $m, l \in N_0$.

There are “generic” complete sets for these classes, similar to those for the $\beta_kP$. Other complete problems are obtained from NP-complete problems by bounding the size of allowable witnesses for membership. Let $\text{CSat}(k)$ be the set of satisfiable, topologically ordered circuits that have fewer than $k \log n$ inputs, where $n$ is the number of gates in the circuit. (Constants 0 and 1 are not counted as inputs.)

Theorem 4.7 [9] For all $k \in N_0$, $\text{CSat}(k)$ is $\leq^{QL}_m$-complete for $N^kP_1$.

Cai and Chen [12] give variants of the bounded weighted circuit satisfiability problem complete for every $N^mP_1$. Additional complete problems for $N^4P_1$ can be obtained by the “forcing” or “greedy” technique of Farr [17] and Szelepcsényi [39] (see Chapter 3) by restricting the amount of nondeterminism to $k \log n$ bits instead of $O(\log^k n)$. For example, the problem $\text{ShortSat}_{ord}(k)$ of Buss and Goldsmith [9] could have been obtained from $\beta_k$-$\text{SAT}$ in this way. (Historically, however, it derived from Abrahamson et al. [1], who were the first to apply the technique to Boolean formulae.)

Farr’s [17] reductions from $\beta_k$-$\text{SAT}$ to restricted versions of the 3-colourability problem or Hamiltonian cycle problem use quasilinear time. Consequently, one can define ordered versions of these problems that are quasilinear-time equivalent to $\text{ShortSat}_{ord}(k)$ and thus are $N^4P_1$-complete. The other reductions considered by Farr and Szelepcsényi use higher-degree polynomials; however, we conjecture that restrictions similar to the ordering property of $\text{ShortSat}_{ord}(k)$ will yield other $N^4P_1$-complete problems. All of these problems can be considered either as restricted versions of NP-complete problems or as generalized versions of P-complete problems.

Not all fixed-parameter versions of NP-complete problems yield complete problems for limited nondeterminism classes. Let $\text{VERTEX-COVER}(k)$ be the set of undirected graphs that have a vertex cover of size $k$. For any fixed $k$, the set $\text{VERTEX-COVER}(k)$ can be accepted in linear time [10, 16].
Chapter 5
Limited Nondeterminism in Other Settings

We give a brief summary of results from Maximization Problems and Automata Theory which are related to limited nondeterminism.

Maximization Problems

In [33], Papadimitriou and Yannakakis defined the complexity classes LOGNP and LOGSNP, which seem to be between the NP-complete sets and P. The purpose of their work was to find a complexity class for which the Vapnik-Chervonenkis Dimension problem is complete. These classes are in direct analogy with the better-known classes, MAXNP and MAXSNP [32]. (For instance, it is known that problems in MAXSNP do not have polynomial-time approximation schemes unless P = NP; it is not yet known whether the optimization versions of LOGNP and LOGSNP have good approximation algorithms.)

Definition 5.1 [33] LOGNP₀ is the class of problems which can be described as:

\[ \{ I : \exists S \in [n]^\log n \forall x \in [n]^p \exists y \in [n]^q \forall j \in [\log n] \phi(I, s_j, x, y, j) \} \]

where \( I \subseteq [n]^m \) is the input relation, \( x \) and \( y \) are tuples of first-order variables ranging over \( [n] = \{1, 2, \ldots, n\} \), and \( \phi \) is a quantifier-free first-order expression involving the relation symbols \( I \) and \( S \), and the variables in \( x \) and \( y \).

LOGSNP₀ is the class of problems which can be described as:

\[ \{ I : \exists S \in [n]^\log n \forall x \in [n]^p \exists y \in [\log n] \phi(I, s_j, x, y, j) \} \]

LOGNP (resp. LOGSNP) is defined as the closure of LOGNP₀ (resp. LOGSNP₀) under polynomial-time many-one reductions.

Definition 5.2 [33] Let \( \mathcal{C} \) be a family of subsets of some universe \( U \). The V-C Dimension of \( \mathcal{C} \), \( d(\mathcal{C}) \), is the largest cardinality of a subset \( S \) of \( U \) such that the following holds: for all subsets \( T \) of \( S \) there is a set \( C_T \in \mathcal{C} \) such that \( S \cap C_T = T \). That is, all subsets of \( S \) are required to be present in \( \mathcal{C} \).

Papadimitriou and Yannakakis [33] showed that V-C Dimension is complete for LOGNP w.r.t. polynomial-time many-one reductions. Cai and Chen [12] characterized LOGNP as the closure of \( \bigcup_c \text{sGC}(c \log^2 n, \Pi^B_k) \) under polynomial-time many-one reductions, where \( \Pi^B_k \) is the class of languages accepted by log-time alternating Turing machines which begin
with universal states and make at most \( k \) alternations. Since \( \Pi^B_k \subseteq P \), it follows from the characterization of \( \beta_2 P \) by [12] (see page 3), that \( \text{LOGNP} \subseteq \beta_2 P \), and thus \( P \subseteq \text{LOGSNP} \subseteq \text{LOGNP} \subseteq \beta_2 P \).

[33] lists a number of problems which are in \( \beta_2 P \), including TOURNAMENT DOMINATING SET, \( \text{Log}^2 \text{Sat} \), and SPARSE SAT. However, Cai and Chen’s result above makes it seem unlikely that V-C DIMENSION is hard enough to be complete for \( \beta_2 P \): the \( \Pi^B_3 \) verifier is very weak (probably weaker than \( P \)). If V-C DIMENSION were complete for \( \beta_2 P \), we would expect to need a polynomial-time verifier. (For a discussion of the relationship of these classes to classes defined by the Guess-and-Gheck model, see [13] and [11].)

Finite Automata

We conclude with a brief look at limited nondeterminism for finite automata and pushdown automata. For finite automata, the nondeterminism considered in this case is not measured in terms of the time needed for an automaton to compute, but rather in terms of the amount of nondeterminism needed to significantly compress a given deterministic finite automaton.

At the lowest end of the computational spectrum lie finite automata. With or without nondeterminism, finite automata accept exactly the class of regular languages; nondeterminism can – in certain cases – greatly reduce the number of states needed to recognize a language [27, 20]. (The amount of nondeterminism in an automaton is strongly related to its degree of ambiguity [21].)

**Definition 5.3** A finite automaton over the alphabet \( \Sigma \) is a 5-tuple \( A = (Q, \Sigma, \delta, q_0, F) \), with \( Q \) a finite set of states, \( q_0 \in Q \), \( F \subseteq Q \), and \( \delta \) a function from \( Q \times \Sigma \) to \( 2^Q \). A move of \( A \) is a triple \( \mu = (p, a, q) \) in \( Q \times \Sigma \times Q \) with \( q \in \delta(p, a) \).

If \( A \) is a finite automaton such that \( |\delta(p, a)| \leq 2 \) for all states \( p \in Q \) and all \( a \in \Sigma \), we define \( \gamma_A(n) \) as the least upper bound on the number of nondeterministic moves needed to accept inputs of length \( n \). It is not hard to see that there exists an automaton \( A' \) such that \( \gamma_{A'}(n) \in \Theta(n) \). Both Simon [38] and Goldstine, Leung, and Wotschke [21] have shown that for any integer \( p \geq 1 \) there exists an automaton \( A'' \) such that \( \gamma_{A''}(n) \in \Theta(n^{1/p}) \). Goldstine, Leung, and Wotschke [21] also show that these sublinear automata have an infinite degree of ambiguity. That is, we cannot bound the number of distinct computations which accept the same string.

The amount of nondeterminism in an NFA is also related to the minimal number of states in a DFA which accepts the same language. Meyer and Fischer [30] showed that for
every $k \geq 1$ there exists a language $R_k$ which is recognized by a $k$-state NFA, but every DFA which recognizes $R_k$ requires at least $2^k$ states. (This is the expected result from the standard subset-construction of a DFA from an NFA.) Kintala and Wotschke [27] showed that, for any $k$-state NFA $A$, if $\gamma_A(n) \leq \log k$ for all $n$, then there is a DFA which accepts the same language, but with a subexponential increase in the number of states.

Kintala and Wotschke [27] also showed that there exist infinite and coinfinite regular languages such that nondeterminism doesn’t help, i.e., the NFA and DFA need the same number of states.

Analogous work on the nondeterminism complexity of PDAs is surveyed by Salomaa and Yu in [36]. As for finite automata, there is a proper hierarchy of sets recognized by machines with increasing degrees of nondeterminism [35, 40], as measured by the depth of the directed acyclic graphs that represent the automata.

In addition, Salomaa and Yu [36] consider the number of computational steps used in a computation. If one defines a measure based on the maximum number of nondeterministic steps of computations on inputs of length $n$, one gets a 3-level hierarchy of complexity. If one considers the minimum number of nondeterministic steps, and takes the max over all strings of length $n$, less is known about the ensuing hierarchy.
References


21


VITA

I. PERSONAL DATA
Name: Matthew Asher Levy
Date of Birth: March 1, 1968
Place of Birth: New York, NY

II. EDUCATION
YEAR DEGREE INSTITUTION
1985–86 Brandeis University, Waltham, MA
1986 University of Maine, Orono, ME
1989–1995 B.S. University of Southern Maine, Portland, ME
1995–2000 University of Kentucky, Lexington, KY

III. PROFESSIONAL POSITIONS
YEAR POSITION COMPANY
2002–present Software Developer IBM, Lexington, KY
2000–2001 I/T Specialist IBM, Lexington, KY

IV. HONORS AND AWARDS
1996, Howard Kurtz Memorial Scholarship
1996–1997, University of Kentucky Presidential Fellowship
1997–1998, University of Kentucky Open Competition Fellowship
1998–1999, University of Kentucky Presidential Fellowship

V. PUBLICATIONS
