Beyond NP: The Work and Legacy of Larry Stockmeyer

Lance Fortnow
University of Chicago
Department of Computer Science
1100 E. 58th. St.
Chicago, Illinois 60637
fortnow@cs.uchicago.edu

> The complexity of algorithms tax even the resources of sixty billion gigabits—or of a universe full of bits; Meyer and Stockmeyer had proved, long ago, that, regardless of computer power, problems existed which could not be solved in the life of the universe.
>
> Frederic Pohl, *Beyond the Blue Event Horizon* [45]

ABSTRACT

Shortly after Steven Cook and Richard Karp showed the existence of many natural NP-complete languages, researchers started to realize the great importance of the P versus NP problem and the difficulty of settling it. One graduate student at the Massachusetts Institute of Technology started to look beyond NP, asking what problems have a higher complexity and how do we classify them. Larry Stockmeyer discovered an amazing structure of complexity classes that continues to direct the research in complexity to this day. Stockmeyer passed away on July 31, 2004 at the age of 55 and in this paper we review some of his research and the legacy he has left on the community.

Categories and Subject Descriptors

K.2 [**History of Computing**]: [Theory, People];
F.1.3 [**Complexity Measures and Classes**]: [Complexity Hierarchies, Reducibility and Completeness]

General Terms

Theory

Keywords

Larry Stockmeyer, Polynomial-Time Hierarchy, Alternation

STOC'05, May 22-24, 2005, Baltimore, Maryland, USA.

1. INTRODUCTION

Larry Stockmeyer started graduate school right at the beginning of the P versus NP era. While many computer scientists started to grapple with the importance and inherent difficulty of that question, Stockmeyer started to ask "What's next?". This retrospective follows Stockmeyer's early research journeys towards finding answers to that question.

In Section 2 we look at how Stockmeyer with his advisor Albert Meyer searched for natural problems that one could prove cannot have efficient solutions. They show the equivalence problem for regular expressions with squaring is EXPSPACE-complete, which implies no polynomial-time (or even polynomial-space) algorithm can determine equivalence.

Stockmeyer and Meyer building on Meyer's lower bounds for a logical theory of numbers EWS1S, show that for some specific reasonable-size inputs, any computer that could determine true formulas for those inputs could not fit inside the known universe.

Meyer and Stockmeyer also attempted to classify the language MINIMAL consisting of formulas not equivalent to any smaller formula. These attempts led to the development of the polynomial-time hierarchy, which we recount in Section 3. Work by Stockmeyer, Meyer and Celia Wrathall give several different characterizations of the hierarchy and show exactly how it could collapse.

Meyer and Stockmeyer give complete sets for the levels of the polynomial-time hierarchy (Section 3.1) and generalize this notion to create a PSPACE-complete set based on quantified Boolean formulas. This result sets the stage for the classic work of Ashok Chandra, Dexter Kozen and Stockmeyer on alternation (Section 4). Chandra, Kozen and Stockmeyer show that alternating time gives you the equivalent space class and that alternating space gives you an exponentially larger time class.

The work on alternation leads to new ways to find natural complete problems for polynomial space and exponential time (Section 4.1), which continues Meyer and Stockmeyer's original search for hard natural problems. Chandra and Stockmeyer exhibit a combinatorial game where it is EXP-complete to determine the winner. This led to researchers finding a number of natural games whose winners cannot be determined in polynomial time.

In Section 5 we look at how Stockmeyer used the polynomial-time hierarchy to approximately count the number of solutions to NP problems. This work led to uniform sampling with an NP oracle and helped us understand the power of several complexity classes.

To understand the legacy of Stockmeyer's work, one needs only to pick up any textbook on computational complexity. Much of complexity builds on one or more ideas originally due to Stockmeyer and his colleagues. In Section 6 we review a few of the many areas where Stockmeyer's research has set the direction of future research in computational complexity.

I focused this retrospective on a few major themes in Stockmeyer's research: the search for hard natural problems, the polynomial-time hierarchy and alternation and approximate counting within the hierarchy. No survey of Stockmeyer's work can adequately cover all of his work in computational complexity and theoretical computer science. In addition to the work presented in this paper, Stockmeyer has had important research in areas like automata theory and parallel and distributed computing that I could not do justice to in this space.

Larry Stockmeyer had the wisdom to look beyond the P versus NP question and found a beautiful structure and amazing theorems that has allowed us to better understand the power and limitations of efficient computation. Hopefully Stockmeyer has moved to a place where all the secrets of complexity are revealed while the rest of us continue to pursue Stockmeyer's visions.

2. HARD NATURAL PROBLEMS

In 1971, Cook [16] showed the question of Boolean formula tautology was hard for every problem in NP[1]. His proof gives a reduction from an arbitrary NP machine to Boolean formula satisfiability. Levin [40] independently proved similar results but these results were not widely known outside of Russia until years later.

In 1972, Karp [34] defined NP-completeness as we know it today and showed that that a number of well-known combinatorial problems like colorability and clique were NP-complete. This started an industry in showing problems NP-complete as illustrated in the 1979 book of Garey and Johnson [24].

Quickly the importance and difficulty of the P versus NP problem became evident. Larry Stockmeyer, then a student at MIT, and his advisor Albert Meyer started asking: if we cannot show that NP-complete problems do not have efficient algorithms what problems can we show cannot be solved quickly?

Hartmanis and Stearns in their seminal paper in computational complexity [28] showed how to construct computable languages using diagonalization that cannot be solved in any fixed time bound. These languages were quite artificial, basically simulating machines with the smaller time bound and doing the opposite. No one had yet came up with a natural example of a computable hard language.

In 1972, Meyer and Stockmeyer [42] found such a problem, the equivalence of regular expressions with squaring. Recall that a regular expression consists of the operations + (union), · (concatenation) and * (Kleene closure) over some finite alphabet Σ, for example $(0+1)^*\cdot 0\cdot 0\cdot(0+1)^*$ represents the set of strings containing two consecutive zeros. Meyer and Stockmeyer looked at a variation that also allows the squaring operation, $R^2 = R \cdot R$, where we could now write the previous expression as $(0+1)^* \cdot 0^2 \cdot (0+1)^*$. Regular expressions with squaring have no more expressive power than regular expressions but can often be written more succinctly, especially if squaring is used recursively, for example expressing the singleton set $\{0^{128}\}$ as $((((((0^2)^2)^2)^2)^2)^2)^2$.

Meyer and Stockmeyer define a language RSQ as the set of regular expressions with squaring not equivalent to Σ^*. They show how to reduce any exponential space computation to an RSQ question and thus any algorithm that solves RSQ must require space (and thus time) exponential in n.

In a future paper [61] Meyer and Stockmeyer would characterize the complexity of equivalence and membership of several other variations of regular expressions.

While Meyer and Stockmeyer's result on RSQ implies exponential time computation in the limit, Stockmeyer wanted to show a problem with a strong lower bound for a specific size. For that he turned to the EWS1S problem, the true sentences in the theory of WS1S, the weak monadic second-order theory of the natural numbers and successor. Meyer [43] had shown that no algorithm solving EWS1S has an elementary recursive time bound.

In a result first appearing in Stockmeyer's thesis [57] and published in J. ACM nearly three decades later [62], Stockmeyer and Meyer do a careful analysis and show that any circuit computing EWS1S on inputs of size 616 would require a circuit of more than 10^{123} gates. Why show such a bound? From Stockmeyer's Ph.D. thesis [57]:

> Thus if a circuit C accepts EWS1S restricted to sentences of length not exceeding 616, and if each gate is the size of a proton, then to accommodate C the entire known universe would be packed with gates.

Stockmeyer assumed a radius of the universe of only 11 billion light years. Current estimates [11] give a radius of 78 billion light years. Using a formula from the thesis we can recompute the lower bound so now you will need sentences of 619 characters to require packing the universe with proton-sized gates.

Based on work on alternation, Stockmeyer and Chandra [60] show certain combinatorial games are EXP-complete and thus one cannot determine the winner efficiently. More in Section 4.1.

3. POLYNOMIAL-TIME HIERARCHY

Meyer and Stockmeyer's paper [42] entitled "The Equivalence problem for Regular Expressions with Squaring Requires Exponential Space" appeared in the 1972 Symposium on Switching and Automata Theory (SWAT), a conference which later became the Symposium on Foundations of Computer Science (FOCS). The paper would go on to become one of the most influential papers in complexity, but not so much for the title result.

Meyer and Stockmeyer also considered the complexity of another natural problem MINIMAL consisting of the set of Boolean formulas for which there is no shorter equivalent formula. Neither MINIMAL nor its complement appear to sit in NP but we can test $\overline{\text{MINIMAL}}$ in NP with an "oracle" for testing equivalence (or non-equivalence) of formulas. The non-equivalence problem is also in NP so we can solve $\overline{\text{MINIMAL}}$ in NP^{NP}. This suggests a "hierarchy" of classes above NP and thus the polynomial-time hierarchy was born.

In Cook's paper [16] on NP-completeness he used a reduction from one language A to a language B by a polynomial-time machine that could make arbitrary queries to B. Such

[1]See Aaronson's Complexity Zoo [1] for full definitions and references for complexity classes mentioned in this survey.

reductions are now called Cook or Turing reductions. Baker, Gill and Solovay [7] formalized a notion of oracle considering classes like P^A and NP^A as problems computable in polynomial-time with access to an oracle A, i.e., they have the ability to make arbitrary adaptive queries written on a special oracle tape that go into a special yes/no state depending on whether the query sits in A.

Meyer and Stockmeyer [42] predate Baker-Gill-Solovay and have to define a nondeterministic polynomial-time reduction. Stockmeyer's later paper on the polynomial-time hierarchy [58] makes use of the Baker-Gill-Solovay setup and we will review the definition of the hierarchy from that paper.

For a class $\mathcal{C}$ we define $\mathrm{P}^{\mathcal{C}}$ and $\mathrm{NP}^{\mathcal{C}}$ as

$$\mathrm{P}^{\mathcal{C}} = \bigcup_{A \in \mathcal{C}} \mathrm{P}^A \qquad \mathrm{NP}^{\mathcal{C}} = \bigcup_{A \in \mathcal{C}} \mathrm{NP}^A$$

We can now define the polynomial-time hierarchy classes Σ_k^p, Π_k^p and Δ_k^p inductively as follows:

- $\Sigma_0^p = \Pi_0^p = \Delta_0^p = \mathrm{P}$,
- $\Delta_{k+1}^p = \mathrm{P}^{\Sigma_k^p}$,
- $\Sigma_{k+1}^p = \mathrm{NP}^{\Sigma_k^p}$ and
- $\Pi_{k+1}^p = \mathrm{coNP}^{\Sigma_k^p}$

Note $\Delta_1^p = \mathrm{P}$, $\Sigma_1^p = \mathrm{NP}$, $\Pi_1^p = \mathrm{coNP}$ and $\Pi_k^p = \mathrm{co}\Sigma_k^p$, $\Delta_k^p \subseteq \Sigma_k^p \cap \Pi_k^p$ and $\Sigma_k^p \cup \Pi_k^p \subseteq \Delta_{k+1}^p$.

The Meyer-Stockmeyer paper first defined the hierarchy and proved that if $\Sigma_k^p = \Delta_k^p$ then $\Sigma_j^p = \Pi_j^p = \Delta_j^p$ for all $j \geq k$. In particular if P = NP then every level of the hierarchy equals P.

Stockmeyer [58] and Wrathall [70] had papers appearing back to back in *Theoretical Computer Science* in 1977 both going more in depth on the polynomial-time hierarchy. These papers mostly extend the ideas and proofs from the Meyer-Stockmeyer paper. They do give a second characterization of the Σ_k^p classes.

THEOREM 1. *L is in Σ_k^p if there is is a k-ary polynomial-time computable relation R such that for all x in Σ^*,*

$$x \in A \Leftrightarrow \exists y_1 \forall y_2 \ldots Q_k y_k R(x, y_1, \ldots, y_k)$$

where the quantifiers alternate and the y_i's range over all strings bounded in length by some fixed polynomial in $|x|$.

This characterization comes from the B_k sets of Meyer-Stockmeyer and formally proved by Wrathall [70].

They also note that $\mathrm{PH} = \cup_{k \geq 1} \Sigma_k^p$ is contained in PSPACE and if PH = PSPACE then $\mathrm{PH} = \Sigma_k^p$ for some k.

Stockmeyer's later work on alternation leads to another characterization of the polynomial-time hierarchy as we will see in Section 4. Stockmeyer also gives another characterization of the hierarchy using second-order predicate calculus building on ideas of Fagin [20].

3.1 Complete Sets in the Hierarchy

Meyer and Stockmeyer [42] give complete sets for the levels of the hierarchy, showing that for every k the set B_k of true quantified Boolean formula starting with $\exists$ and having $k-1$ alternations of quantifiers is Σ_k^p-complete. B_1 has just existential quantifiers and is the same as SAT.

> We do not claim that the languages B_k are natural problems. However they may provide a useful intermediate step in exhibiting natural problems which are complete in some class above NP, just as Cook's proof of the $k = 1$ case provided a handle on the class NP. [42, p. 128]

Stockmeyer [58] gives a slightly more "natural" Σ_2^p-complete set N-INEQ, the inequivalence of expressions over sets of integers built up by union and sums.

Only in the past decade have we really started to see a number of natural problems complete for the second and third levels of the polynomial-time hierarchy (see [51]). For example, Succinct Set Cover is Σ_2^p-complete [66] and VC Dimension (with sets described by a circuit) is Σ_3^p complete [50]. As predicted by Meyer and Stockmeyer all of these results are proven from direct or indirect reductions from B_2 and B_3.

The actual complexity of MINIMAL remains open for circuits or Boolean formula. Umans [67] shows the following related problem is Σ_2^p-complete: Given a DNF formula ϕ and an integer k, is there a formula ψ equivalent to ϕ of size at most k?

4. ALTERNATION

Stockmeyer and Meyer [61, 58] define a set they called the ω-jump of the polynomial-time hierarchy as

$$B_\omega = \bigcup_{k \geq 1} B_k$$

in an analogy to the ω-jump of the arithmetic hierarchy.

B_ω is just the set of true quantified Boolean formula with an arbitrary number of quantifiers, a set we now call TQBF or just QBF.

Stockmeyer and Meyer show that B_ω is PSPACE-complete even when restricted to a 3CNF in the formula. Their proof uses techniques from Savitch's Theorem [49] that showed how to simulate nondeterministic space $s(n)$ in deterministic space $s^2(n)$.

Stockmeyer and Meyer didn't realize it at the time but they discovered a whole new way to view space complexity using alternation, an idea formalized in the now classic 1981 paper [14] on alternation by Ashok Chandra, Dexter Kozen and Stockmeyer that combined the results of two 1976 FOCS papers [15, 39].

A nondeterministic machine should really have the name "existential machine" since it accepts if there exists an accepting computation. One can also imagine a universal machine that accepts if all of its computation paths accept. One can then consider machines that alternate between existential and universal quantification. Chandra, Kozen and Stockmeyer formally define these alternating Turing machines and prove some amazing connections to deterministic time and space classes.

THEOREM 2 (CHANDRA-KOZEN-STOCKMEYER). *For reasonable $s(n) \geq \log n$ and $t(n) \geq n$,*

1. $\mathrm{NSPACE}(s(n)) \subseteq \mathrm{ATIME}(s^2(n))$,
2. $\mathrm{ATIME}(t(n)) \subseteq \mathrm{DSPACE}(t(n))$ *and*
3. $\mathrm{ASPACE}(s(n)) = \cup_{c>0}\mathrm{DTIME}(c^{s(n)})$.

where ATIME and ASPACE are the alternating analogues of DTIME and DSPACE.

If we stick an A in front of a class name to get the alternating version, Theorem 2 gives us equivalences to deterministic classes like AP = PSPACE, APSPACE = EXP and AL = P. As Chandra, Kozen and Stockmeyer [14] put it: The deterministic hierarchy

$$\mathrm{L} \subseteq \mathrm{P} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{EXPSPACE} \subseteq \ldots$$

shifts by exactly one level when alternation is introduced.

Alternation has application to the earlier themes in Stockmeyer's research. Chandra, Kozen and Stockmeyer [14] give another characterization of the polynomial-time hierarchy. We define a Σ_k machine as one that has k layers of alternations starting with existential. For example a Σ_2-machine has existential configurations and then universal configurations. We can define Π_k machines the same way except it starts with universal configurations.

THEOREM 3 (CHANDRA-KOZEN-STOCKMEYER). *For all k, the class Σ_k^p is exactly the set of languages accepted by polynomial-time Σ_k machines and Π_k^p is exactly the set of languages accepted by polynomial-time Π_k machines.*

The proof of Theorem 3 is by induction on k similar to the proof of Theorem 1.

They also define a similar hierarchy for log-space. Seven years later Immerman [30] and Szelepcsényi [63] would prove NL = coNL and thus the whole log-space hierarchy collapses to NL.

Chandra, Kozen and Stockmeyer [14] also formally define alternating finite automata and show they accept the same set as deterministic automata but with a possible double exponential blowup in the number of states. Given an alternating automata one defines an equivalent deterministic automata by having a state for each function from the original sets of states to $\{0, 1\}$. They also create a language over a three-letter alphabet that has a k-state alternating automata but any deterministic automata requires 2^{2^k} states.

4.1 Complete Sets for PSPACE and EXP

While computer scientists discovered many NP-complete problems in the early 70's (see Garey and Johnson [24]), they found very few natural problems complete for PSPACE and EXP. Stockmeyer and Meyer [61, 58] gave a PSPACE-complete problem B_ω (now TQBF) by generalizing the complete sets developed by Meyer and Stockmeyer [42] for levels of the polynomial-time hierarchy. While most PSPACE-hardness results are proven from direct or indirect reductions from TQBF it was the work on alternation by Chandra, Kozen and Stockmeyer [14] that developed the intuition for understanding PSPACE-hardness results for many natural problems.

We can view the alternating Turing model of Chandra, Kozen and Stockmeyer as a game. For example consider the following setup: We have two players who alternate writing down bits for some polynomial number of steps. A fixed polynomial-time computable judge looks at the bits written and determines a winner. Determining whether the first player has a winning strategy is equivalent to an alternating Turing machine where player 1 plays the existential alternatives and player 2 plays the universal. So by Chandra-Kozen-Stockmeyer we have that determining the winner of the game is PSPACE-complete.

This view helped produce a large number of natural PSPACE-hard problems usually from reductions from TQBF. Even and Tarjan [19] gave an early example, showing a generalization of Hex is PSPACE-complete. Schaefer [52] showed a number of other games, like Generalized Geography and Kayles, are also PSPACE-complete.

Let's consider a popular perfect information game like checkers. Checkers is played on 8×8 board so there are only a finite number of possible positions and the computational complexity question is not interesting. Instead you can generalize checkers to an $N \times N$ board in a reasonable way. Fraenkel, Garey, Johnson, Schaefer and Yesha [22] showed that generalized checkers is PSPACE-hard. Generalized Checkers would be PSPACE-complete if the the number of turns is bounded by a polynomial in the board size but the rules of checkers do not require this.

Stockmeyer and Chandra [60] show that one can use the Chandra-Kozen-Stockmeyer [14] characterization of EXP as alternating polynomial space to show that some games are EXP-complete and thus provably do not have an efficient algorithm to determine the winner. Stockmeyer and Chandra give an example of such a combinatorial game. Many researchers used the tools of Stockmeyer and Chandra to show the EXP-completeness of many natural games. In 1984, Robson [46] showed that generalized checkers is indeed EXP-complete.

Eppstein maintains a web page [18] surveying the computational complexity of many common games, several of which are PSPACE and EXP complete.

5. APPROXIMATE COUNTING

In 1979 Valiant [68] considered the problem of computing a permanent of a matrix, which for zero-one matrices is equivalent to counting the number of perfect matchings in a bipartite graph. Valiant defined the counting class #P containing the functions f such that $f(x)$ is the number of accepting computations of $M(x)$ for some NP machine M. Valiant showed that the permanent as well as many other related problems are #P-complete [68, 69].

In particular Valiant's result meant that computing the permanent was as hard as counting the number of satisfying assignments of a Boolean formula. Many researchers felt that if we couldn't compute the permanent exactly, perhaps we can get a good approximation. Early algorithms had either high variances or long running times (see [59] for discussion).

Stockmeyer tackled a broader question of approximating any counting problem, i.e., approximating a #P function. Building on ideas of Sipser [56], Stockmeyer showed that one can approximate the permanent within the polynomial-time hierarchy.

THEOREM 4 (STOCKMEYER). *For any #P function f and every polynomial p there exists a function g computable in polynomial-time with a Σ_2^p oracle such that*

$$f(x) \leq g(x) \leq (1 + 1/p(|x|))f(x)$$

A randomized algorithm only needs an NP *oracle to achieve the approximation with high probability.*

Stockmeyer also gave a relativized world where one could not deterministically approximate an arbitrary #P function.

Stockmeyer noted that approximating the number of satisfying assignments would mean you could solve satisfia-

bility, and so approximating arbitrary #P functions is NP hard. You can check in polynomial time whether the permanent of a 0-1 matrix is positive (this is just bipartite matching) so Stockmeyer asked whether approximating the permanent is NP-hard. Recently Jerrum, Sinclair and Vigoda [31] found an efficient probabilistic algorithm for approximating the permanent using rapidly-mixing Markov chains. Thus approximating the permanent is not NP-hard unless NP is in BPP and the polynomial-time hierarchy collapses.

Stockmeyer's result took on greater importance when a few years later Toda [64] showed that every language in the polynomial-time hierarchy reduces to #P. Thus computing #P functions exactly are much harder than approximating them. If we can compute #P-complete functions like the permanent exactly within the polynomial-time hierarchy then the hierarchy would collapse.

Jerrum, Valiant and Vazirani [32] and Bellare, Goldreich and Petrank [8] give a randomized equivalence between approximately counting the size of polynomial-time computable sets and uniformly generating elements of such sets. So Stockmeyer's result now allows us to uniformly generate elements of an easily computable set using an NP oracle.

Bshouty, Cleve, Gavaldà, Kannan and Tamon [12] use uniform generation in their paper showing how to learn circuits with an NP oracle and equivalence queries. Köbler and Watanabe [38] use Bshouty et. al. to show that if NP has polynomial-size circuits then the polynomial-time hierarchy collapses to $\mathrm{ZPP}^{\mathrm{NP}}$, i.e., probabilistic algorithms with an NP oracle that run in expected polynomial-time and never err. This improves the original collapse due to Karp and Lipton [35].

Cai [13] uses approximate counting to show that S_2^p, a class defined by Russell and Sundaram [47], is contained in $\mathrm{ZPP}^{\mathrm{NP}}$. Cai's paper shows that if NP has polynomial-size circuits then the polynomial-time hierarchy collapses to S_2^p, the strongest known collapse.

Shaltiel and Umans [53] improving on Klivans and van Melkebeek [36] show how to derandomize Stockmeyer's approximate counting under reasonable assumptions.

6. LEGACY

Stockmeyer's work permeates much of computational complexity. Complexity theorists use the models, theorems and techniques to help understand everything from circuit complexity to interactive proof systems. In this section we describe the effect of Stockmeyer's research on a few of the many areas in recent research in complexity.

6.1 Circuit Complexity

One way we can measure the complexity of a Boolean function is by studying the size of the circuits that compute that function, sometimes with restrictions on the gate fan-in and the depth. For example Stockmeyer and Meyer's lower bound for EWS1S [57, 62] mentioned in Section 2 shows a lower bound on the number of gates with fan-in 2 needed by a circuit to solve the problem.

Circuit complexity became popular in the mid-1980's as a potential way to attack the P versus NP problem (see [10]) but much of the circuit research actually came out of questions related to the polynomial-time hierarchy.

In the seminal paper on relativization, Baker, Gill and Solovay [7] leave open whether there exists an oracle relative to which the polynomial-time hierarchy is infinite, i.e., an oracle A where $\Sigma_k^{p,A} \neq \Sigma_{k+1}^{p,A}$ for all k. Sipser [55] showed that if one could prove that a certain class of functions F_d required large enough $(d-1)$-depth circuits then one would have the desired oracle. Ajtai [2] and Furst, Saxe and Sipser [23] proved nontrivial lower bounds on constant-depth circuits but not strong enough for the oracle. Yao first proved a strong enough lower bound in a paper [71] entitled "Separating the Polynomial-Time Hierarchy by Oracles." Thus modern circuit complexity was born by finding oracles for the polynomial-time hierarchy.

Håstad [29] developed a "switching lemma" to get nearly tight bounds on constant-depth circuits. Ko [37] using careful applications of the switching lemma showed that the polynomial-time hierarchy could collapse at any level in some relativized world: For every $k \geq 0$ there were two oracles A_k and B_k such that relative to A_k,

$$\Sigma_k^p \neq \Sigma_{k+1}^p = \Sigma_{k+2}^p = \mathrm{PSPACE}$$

and relative to B_k,

$$\Sigma_k^p \neq \Sigma_{k+1}^p = \Sigma_{k+2}^p \neq \mathrm{PSPACE}.$$

Ruzzo [48] used alternating Turing machines based on Chandra-Kozen-Stockmeyer [14] to define uniform circuit classes. For example uniform constant depth circuits (AC_0) are defined by alternating log-time machines with a constant number of alternations.

6.2 Infinite Hierarchy Conjecture

We say the polynomial-time hierarchy is infinite if $\Sigma_k^p \neq \Sigma_{k+1}^p$ for all $k > 0$ and otherwise we say it collapses. While Stockmeyer has often stated this as an open problem and notes that the hierarchy separates in the arithmetic hierarchy [58], to the best of my knowledge he has never conjectured whether or not the polynomial-time hierarchy is infinite.

Given the lower bounds on constant-depth circuits by Yao [71] and Håstad [29] and their connection to the polynomial-time hierarchy mentioned in Section 6.1, many complexity theorists do now conjecture the hierarchy is infinite and such a conjecture comes in quite useful in that it implies many other conjectures in computational complexity. We see many results, sometimes called "pigs can fly" theorems, that show that if some conjecture does not hold then the polynomial-time hierarchy collapses. When someone eventually does prove that the polynomial-time hierarchy is infinite then we will immediately get that all these conjectures are true. In this section we give a small sample of these kinds of results.

Karp and Lipton [35] show that if NP-complete problems have polynomial-size circuits then the polynomial-time hierarchy collapses. This gives evidence that NP-complete problems do not have polynomial-size circuits and led to the approach of proving P $\neq$ NP by trying to show super-polynomial lower bounds for circuits for NP-complete problems.

One can define a Boolean Hierarchy above NP as follows: $\mathrm{BH}_1 = \mathrm{NP}$ and BH_{k+1} is the set of languages that are the set difference of B and C for some B in NP and C in BH_k. Kadin [33] shows that if the Boolean hierarchy collapses then the polynomial-time hierarchy collapses, giving evidence that the Boolean hierarchy is infinite.

Babai [5, 6] defined the class AM (Arthur-Merlin) of problems that can be solved by verification of probabilistically generated questions. Results from Goldreich, Micali and

Wigderson [25] and Goldwasser and Sipser [27] show that Graph Non-Isomorphism sits in AM. Boppana, Håstad and Zachos [9] show that if co-NP is in AM then the polynomial-time hierarchy collapses.

Putting these all together we get that if the polynomial-time hierarchy is infinite then Graph Isomorphism is not NP-complete, the best evidence we have that Graph Isomorphism is inherently easier than problems like Boolean-Formula Satisfiability.

6.3 Interactive Proofs

Papadimitriou [44] developed a model of *Games Against Nature* where he looks at alternating polynomial-time like Chandra-Kozen-Stockmeyer [14] except that instead of a universal player we have a player that just sends independent uniform random bits. A string is in the language if the first player wins with probability at least one half. Papadimitriou shows this model still captures exactly PSPACE.

Babai's Arthur-Merlin model [5, 6] is essentially the same as Papadimitriou's Games Against Nature except that the acceptance probability must be bounded away from one half. Goldwasser and Sipser [27] showed the model equivalent to an independently developed model of interactive proof system by Goldwasser, Micali and Rackoff [26], which allowed any probabilistic verifier possibly using hidden coins.

Shamir [54] showed how to extend the protocol of Lund et. al. [41] to show every language in PSPACE has an interactive proof, i.e., Papadimitriou's result still holds if the acceptance probabilities are bounded away from one half. Shamir's proof used properties of quantified Boolean formulas and the fact that the set of true QBFs is PSPACE-complete, first proven by Stockmeyer and Meyer [61, 58].

Dwork and Stockmeyer [17] consider interactive proof systems with verifiers, which are 2-way probabilistic finite state automata. They show that the Goldwasser-Sipser [27] result does not hold in this model–private coin proof systems are strictly more powerful than the public coin counterpart. They also show every language in EXP has a private coin proof system.

Interactive proof systems led to an connections to limitations on approximation [21], which led to the theory of probabilistically checkable proof systems [4, 3]. A large number of important papers improved PCPs to prove stronger lower bounds for approximation for specific problems.

7. CONCLUSION

When others asked what can we compute, Stockmeyer started looking at what we can't compute. Along the way he developed the computational tools and models that have guided complexity ever since. We lost one of the true giants of computational complexity last summer but Stockmeyer's legacy will continue to grow with generations of theorists to come.

Acknowledgments

I would like to thank Cynthia Dwork and the STOC program committee for entrusting me with the task of giving this retrospective.

Thanks to Ron Fagin, Bill Gasarch, Nanda Raghunathan and Rahul Santhanam for comments on earlier drafts.

Most of all I would like to thank Larry Stockmeyer. While I haven't discussed much of my own work in this survey, his theorems, definitions and simply his way of viewing complexity classes have played a critical role in nearly all of my research in computational complexity.

8. REFERENCES

[1] S. Aaronson. The complexity zoo. http://complexityzoo.com.

[2] M. Ajtai. Σ^1_1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[4] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

[5] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, New York, 1985.

[6] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[7] T. Baker, J. Gill, and R. Solovay. Relativizations of the P = NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[8] M. Bellare, O. Goldreich, and E. Petrank. Uniform generation of NP-witnesses using an NP-oracle. *Information and Computation*, 163:510–526, 2000.

[9] R. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[10] R. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 14, pages 757–804. North-Holland, 1990.

[11] R. Britt. Universe measured: We're 156 billion light-years wide!, May 2004. Space.com.

[12] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, June 1996.

[13] J. Cai. $S_2^p \subseteq ZPP^{NP}$. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 620–628. IEEE, New York, 2001.

[14] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[15] A. Chandra and L. Stockmeyer. Alternation. In *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science*, pages 98–109. IEEE, New York, 1976.

[16] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158. ACM, New York, 1971.

[17] C. Dwork and L. Stockmeyer. Finite state verifiers I: The power of interaction. *Journal of the ACM*, 39(4):800–828, October 1992.

[18] D. Eppstein. Computational complexity of games and puzzles. http://www.ics.uci.edu/~eppstein/cgt/hard.html.

[19] S. Even and R. Tarjan. A combinatorial problem which is complete in polynomial space. *Journal of the ACM*, 23(4):710–719, October 1976.

[20] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity of computation*, volume 7 of *SIAM-AMS Proceedings*, pages 43–73. Society for Industrial and Applied Mathematics, 1974.

[21] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, March 1996.

[22] A. Fraenkel, M. Garey, D. Johnson, T. Schaefer, and Y. Yesha. The complexity of checkers on an $N \times N$ board. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 55–64. IEEE, New York, 1978.

[23] M. Furst, J. Saxe, and M. Sipser. Parity, circuits and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[24] M. Garey and D. Johnson. *Computers and Intractability. A Guide to the theory of NP-completeness*. W. H. Freeman and Company, New York, 1979.

[25] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[26] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[27] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, Greenwich, 1989.

[28] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[29] J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, 1989.

[30] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[31] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. In *Proceedings of the 33rd ACM Symposium on the Theory of Computing*, pages 712–721. ACM, New York, 2001.

[32] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[33] J. Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM Journal on Computing*, 17(6):1263–1282, December 1988.

[34] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[35] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on the Theory of Computing*, pages 302–309. ACM, 1980.

[36] A. Klivans and D. van Melkebeek. Graph nonisomorhism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[37] K. Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM Journal on Computing*, 18:392–408, 1989.

[38] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. *SIAM Journal on Computing*, 28(1):311–324, February 1999.

[39] D. Kozen. On parallelism in Turing machines. In *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science*, pages 89–97. IEEE, New York, 1976.

[40] L. Levin. Universal'nyĭe perebornyĭe zadachi (Universal search problems: in Russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973. Corrected English translation in [65].

[41] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[42] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129. IEEE, New York, 1972.

[43] L. Meyer. Weak monadic second-order theory of successor is not elementary recursive. In *Proceedings of the Boston University Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 132–154. Springer, 1975.

[44] C. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

[45] F. Pohl. *Beyond the Blue Event Horizon*. Ballantine Books, 1980. Chapter 12.

[46] J. Robson. $N \times N$ checkers is EXPTIME complete. *SIAM Journal on Computing*, 13(2):252–267, May 1984.

[47] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7(2):152–162, 1998.

[48] W. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, pages 365–383, 1981.

[49] W. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[50] M. Schaefer. Deciding the vapnik-chervonenkis dimension is Σ_3^p-complete. *Journal of Computer and System Sciences*, 58:177–182, 1999.

[51] M. Schaefer and C. Umans. Complete in the polynomial-time hierarchy: a compendium. *SIGACT News*, 33(3):32–49, September 2002.

[52] T. Schaefer. On the complexity of some two-person perfect information games. *Journal of Computer and System Sciences*, 16:185–225, 1978.

[53] R. Shaltiel and C. Umans. Pseudorandomness for approximate counting and sampling. In *Proceedings of the 20th IEEE Conference on Computational Complexity*. IEEE Computer Society, Los Alamitos, 2005. To appear.

[54] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[55] M. Sipser. Borel sets and circuit complexity. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 61–69. ACM, New York, 1983.

[56] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 330–335. ACM, New York, 1983.

[57] L. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, Massachusetts Institute of Technology, June 1974.

[58] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[59] L. Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):1–13, November 1985.

[60] L. Stockmeyer and A. Chandra. Provably difficult combinatorial games. *SIAM Journal on Computing*, 8(2):151–174, May 1979.

[61] L. Stockmeyer and A. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th ACM Symposium on the Theory of Computing*, pages 1–9. ACM, New York, 1973.

[62] L. Stockmeyer and A. Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *Journal of the ACM*, 49(6):753–784, November 2002.

[63] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[64] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[65] R. Trakhtenbrot. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[66] C. Umans. Hardness of approximating Σ_2^p minimization problems. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 465–474. IEEE, New York, 1999.

[67] C. Umans. The minimum equivalent DNF problem and shortest implicants. *Journal of Computer and System Sciences*, 63(4):597–611, December 2001.

[68] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[69] L. Valiant. The complexity of reliability and enumeration problems. *SIAM Journal on Computing*, 8:410–421, 1979.

[70] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

[71] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10. IEEE, New York, 1985.