OBSERVATIONS ABOUT THE DEVELOPMENT OF THEORETICAL COMPUTER SCIENCE

J. Hartmanis*

TR80-404

Paper presented at the 20th Annual IEEE Symposium on Foundations of Computer Science, Puerto Rico, 1979.

Computer Science Department
Cornell University
Ithaca, N.Y. 14853

*This work has been supported in part by National Science Foundation Grants MCS 78-00418 and MCS 79-01439.

OBSERVATIONS ABOUT THE DEVELOPMENT OF THEORETICAL COMPUTER SCIENCE

J. Hartmanis*

Department of Computer Science
Cornell University

This paper gives a personal account of some developments in automata theory and computational complexity theory. Though the account is subjective and deals primarily with the research areas of direct interest to the author, it discusses the underlying beliefs and philosophy which guided this research as well as the intellectual environment and the ideas and contacts which influenced it. An attempt is also made to draw some general conclusions about computer science research and to discuss the nature of theoretical computer science.

INTRODUCTION

This paper gives a subjective, historical account of some developments in automata theory and the theory of computational complexity in which I have participated. In this account I describe, besides the developments in these fields, the intellectual environment of this period, our motivation and the ideas and people who influenced us and contributed to these developments. Though the account is subjective and of limited scope, it draws some general conclusions about theoretical computer science and discusses the nature of computer science in general.

My observation and study of computer science for the last twenty years has convinced me that computer science is a new type of science with a potential for immense importance if it continues to develop properly. First of all, it is obvious that the unprecedented technological developments in electronics, which we are witnessing today, are accelerating the already explosive growth of computer applications and they are penetrating almost all aspects of our society. These technological developments are going to have a profound impact on our society, which we can only dimly perceive today, and they give an immense potential importance to computer science. Today, in the full sense of the word, they are buffeting computer science by creating unexpected demands for well educated people as well as creating demands and potential for computer science advances to harness and exploit the possibilities created by the technological developments and guide their applications.

Furthermore, I believe that more broadly computer science can play an important role through the intellectual concepts and tools it generates and their application to other disciplines. I see computer science as a new scientific discipline which is building a completely new set of conceptualizations, theories and applications, that create an intellectual arsenal from which computer scientists and scientists from other disciplines can borrow the intellectual concepts and models to think about their specific problems and build from their own theories.

Relative to other sciences I see computer science as a brand new species among them and I believe that it differs fundamentally from the older sciences. As a matter of fact, I am convinced that in large parts of computer science the classic research paradigms from physical sciences or mathematics do not apply and that we have to develop and understand the new paradigms for computer science research. The fundamental difference between, say, physics and computer science is that in physics, we study to a very large extent a world which exists and the main objective is to observe and explain the existing (and predict new observable) phenomena. The relations between experiments and theory are quite well understood and richly illustrated by successful examples. Computer Science, on the other hand, is primarily interested in what can exist and how to describe and analyze the possibly in information processing. It is a science which has to conceptualize, create the intellectual tools and theories to help us imagine, analyze and build the feasibly possible. Computer Science is indeed a different intellectual discipline than we have ever encountered before. It shows some haunting similarities with physical sciences and mathematics (whose basic research paradigms and goals are quite different), but it differs from both of these disciplines in some very fundamental ways. As a matter of fact, quite often the paradigms, borrowed from physical sciences and mathematics have been incorrectly applied to computer science research with predictably frustrating results.

In view of these observations, I consider that one of the very important tasks for the computer science community is to understand better the nature of their science and develop the new research norms, paradigms and methodology without which it will not mature into an independent and influential science. In particular, the relations between theoretical and experimental computer science must be clarified and new interactions must be forged. This is not just a matter of producing "more practical theories" and applications of theory which we certainly need. It is the hard and challenging task

*This work has been supported in part by National Science Foundation Grants MCS 78-00418 and MCS 79-01439.

of finding out for the first science from a new species of sciences how theory, experiments and practice should interact. Furthermore, this is not just an esoteric exercise in philosophy of science; whether we admit it or not, our underlying beliefs, our conception of our field of study, and our perception of what is possible, all fundamentally influences what kind of science we are going to build.

THE EARLY YEARS

When I had to start thinking about my choices in higher education and an eventual career, Europe was just starting to dig out and recover from the devastation of World War II. The war had not only caused immense physical destruction and human suffering in Europe, it had also destroyed many values, traditions, idealogies and institutions. At this time, while I was stranded in one of the most devastated foreign countries in Europe, I received my first serious exposure to science and had the fortune of meeting some first class research scientists. In this wasteland of human, physical, spiritual and idealogical destruction, science stood for me proudly untouched by the destruction and apparent devaluation of almost everything else.

The appeal of science at this time in this environment and its later influence on me was profound. I still believe that science is one of the most interesting and among the noblest human endeavors, but for many young scientists it is today just one of the many possible interesting careers they could have chosen (and most likely not the one with the highest material rewards). Looking back it still seems that I had almost no other choice but to become a scientist. I immersed myself in the study of physics and the necessary mathematics and was fascinated by the beauty and power of physical theories from classical mechanics and thermodynamics to quantum theory and relativity. I was also impressed by the detached spartan elegance of mathematics which, almost miraculously against its own inner drive towards abstractness, could not escape from repeatedly finding applications.

I was fully exposed to the rich cultural heritage of physics and mathematics with the illustrious history, colorful personalities, and awesome intellectual achievements of these sciences. Ever since this early exposure to science, I have loved since and found in history of science a lot of enjoyment and inspiration.

Compared to the rich intellectual background and proud history of scientific achievement in physics and mathematics computer scientist must still be considered "intellectual orphans." Or maybe one should consider them to be the "intellectual nouveau riches" of science with their implied lack of cultural background and traditions, arrogance and strong desire to be recognized as equals by the intellectual aristocracy of the older sciences.

After concentrating on the study of physics in Europe, I was converted to pure mathematics in this country and I completed my Ph.D. work in lattice theory at the California Institute of Technology in 1955. During this period I met for the first time Johnny von Neumann when he lectured in 1952 at Cal Tech on his work on "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components" [14]. I must admit that at this time I failed completely to understand the importance of von Neumann's work on computing machines and automata theory. His enthusiasm and brillance were obvious, but my indoctrination in pure mathematics was too strong to be penetrated by such ideas at this time. As a matter of fact, many of the Cal Tech mathematicans lamented the sad fact that "Johnny had given up doing real mathematics." In retrospect it is quite a shocking misjudgment, but in the mathematical community it is by far not an isolated incident with respect to computer science.

After von Neumann left we returned to the study of his continuous dimensional geometries which was judged to be "real mathematics."

My first serious exposure to computer science came in the summer of 1957 when Dr. R. L. Shuey convinced me to spend the summer in the Information Studies Section of the General Electric Research Laboratory in Schenectady, New York. This time, removed from my traditional background and exposed to a wealth of new ideas, I quickly realized that something exciting was happening in computing and that there were some very interesting research problems. The summer of 1957 sufficed to convince me that I wanted to work in computer science and I returned to the General Electric Research Laboratory as a research scientist in 1958. I spent the following seven interesting and productive years at the G.E. Research Laboratory until 1965 when I returned to Cornell University to chair the newly established Computer Science Department.

The main research themes to which I was exposed at this time and which, in a sense, dominated the intellectual effort in the early computer science developments were switching theory, finite automata and information theory and coding. During these early days at the General Electric Research Laboratory my thinking was very strongly influenced by Claude Shannon's work. I studied with interest Switching Theory [20] and Communication Theory [21] or Information Theory, as it is often referred to today. The importance of Shannon's work on switching theory was in providing a beatiful example of how important practical problems can be submitted to rigorous mathematical treatment. This was not a theory to explain any natural phenomena, but a mathematical formalization which permitted the analysis and creation of better man-made systems. It also alerted me to the importance of algebraic methods in this area of research.

Already on my first reading I found Shannon's Communication Theory intuitively clear and with beautiful and surprising results. Here was an elegant example of a quantitative theory of (non-physical) laws which govern information transmis-

sion. This had a very profound impression on my thinking and suggested to me that there may be a quantitative theory of computing. My early attempts at a direct extension of information theory to information processing failed. In retrospect, my initial ideas in this research area were embarrassingly naive.

It should be recalled that at this time, during the late fifties, I and my colleagues at the G.E. Research Laboratory knew very little about effective computability and Turing machines. I had received my Ph.D. degree in mathematics from the California Institute of Technology without being exposed to a modern course in logic, which they did not have at that time. Nor can I recall any serious discussion about effective computability during my graduate student days. Surprisingly, as I will explain later, our ignorance in these matters may have turned out to be a blessing in our later research efforts.

After some exploratory work in computer science and some guidance by C. L. Coates and P. M. Lewis, I was attracted to the study of finite automata and concentrated my reading and research effort in this area. I read "Automata Studies" which contained an interesting selection of papers edited by C. E. Shannon and J. McCarthy [22]. From this selection I was particularly influenced by E. F. Moore's "Gedanken-Experiences on Sequential Machines" and S. C. Kleene's "Representation of Events in Nerve Nets and Finite Automata." The collection also contained von Neumann's paper on Probabilistic Logics which I had encountered before. I was furthermore influenced and impressed through personal contacts and the literature by D. A. Huffman and E. J. McClusky.

In my research on finite automata, I started a systematic investigation of the problem of realizing finite automata from interconnections of smaller automata. I believe that the selection of this problem area was partially influenced by Howard Aiken's call for the development of a "switching theory for systems," which he made at the 1958 IFIP Congress. Though the initial results of decomposition of finite automata were suggested by decomposition results from abstract algebra, I soon discovered that automata behaved differently than algebras and that one had to ask different questions and develop new mathematical tools. I was joined in this research effort by R. E. Stearns in 1960 when he came to the G.E. Research Laboratory as a summer employee. A year later, after receiving his Ph.D. from Princeton University, he joined us as a research scientist and we continued our intensive and fruitful collaboration until I left the Laboratory.

The decomposition work led us to some very interesting questions and we developed new mathematical methods and eventually built a structure theory for finite automata. This theory was also applied with considerable success to the problem of assigning efficient codes to the internal states of sequential machines, as described in our papers on this problem [5,25]. Our work in this area culminated in our book "Algebraic Structure Theory for Sequential Machines," which appeared in 1966 [7].

Though there are some very interesting incidents and many lessons to be learned from this work, I will not discuss them here since I want to turn next to the development of computational complexity and discuss that topic in more detail. Still it would be an omission not to mention in this context the related work of K. B. Krohn and J. L. Rhodes on machine decomposition [9].

In the early sixties Krohn and Rhodes studied the semigroups defined by automata and defined decompositions of automata in terms of the semigroups they generated. This theory is mathematically sophisticated and shows that every finite automaton with semigroup G can be realized by a loop-free connection of automata with simple groups which "divided" G and from two-state automata. Conversely, every simple group which "divides" G has to "divide" the semigroup of some component automaton in the decomposition. Unfortunately, the decompositions defined in terms of semigroups do not contain all possible decompositions of an automaton into a loop-free connection of smaller (fewer state) automata. Many automata with simple groups can be decomposed further. The reason for this is easily seen if we consider permutation automata which generate groups. Decompositions are defined by homomorphisms and a group homomorphism is defined by a normal subgroup because in a group we can multiply from the left or the right. In an automaton the input action corresponds to multiplication from only one side and therefore, the homomorphisms of these automata correspond to subgroups of the group they generate (instead of normal subgroups). The decompositions of automata defined by homomorphisms corresponding to subgroups which are not normal are by definition not admitted as decompositions in this theory, though they exist in a strong physical sense. Surprisingly, this fact did not dampen much the original enthusiasm for this theory even if it did not yield the expected decompositions.

COMPUTATIONAL COMPLEXITY

In the early sixties our work on the structure theory for finite automata progressed very rapidly and we decided with Dick Stearns to write an unified exposition of this research area. At the same time we realized that finite automata did not provide us with a sufficiently rich model of computing to develop the quantitative theories which we believed were needed and could be created. We were also becoming more aware of our ignorance about effective computability and Turing machines.

In view of this situation, early in 1962 we started reading about effective computability, Turing machines and formal systems. Our main source was Martin Davis' "Computability and Unsolvability" [4]. We also read Turing's classic paper on Computable Numbers, R. M. Smullyan's "Theory of Formal Systems" [24] and H. Hermes' "Aufzaehlbarkeit, Entscheidbarkeit, Berechenbarkeit" [3]. I suspect that we learned about Smullyan's work from

Myhill's interesting technical report on linear bounded automata [13], to which I will return later.

The spring of 1962 must have been for us an intellectually wild time. In April, while intensively working on finite automata structure theory, we started reading about Turing machines, at the same time we studied the Krohn-Rhodes decomposition theory and we were for the first time seriously exposed to context-free languages and push-down automata through N. Chomsky's report in MIT Electronics Laboratory Quaterly Progress Report [3]. As you can see, our intellectual diet was varied and came from a wide mix of sources.

The beauty and simplicity of the concept of effective computability in the Turing machine formulation impressed us deeply. In a very short time we had grasped the basic ideas and started thinking, at least, for us, new thoughts about these topics. We soon proved our first undecidability result and, motivated by our parallel work on finite automata, proved in July that Turing machine computation time had linear speed-up. These were simple ideas, but they permitted us to gain experience and confidence with these concepts.

We discussed the new ideas in intensive sessions and then returned to reading just enough to help us out of the conceptual difficulties which arose in our discussions. I am sure that our struggle and our early results would have looked quite pathetic to people knowledgeable in this area and that they would have eagerly helped us and most likely directed our attention to the well established problem areas. In a sense we had the good fortune of coming out of our ignoarance in our own way without being embarrassed or intellectually dominated by other people. What we wanted to do at this time was more important for us than what we knew.

On July 12, we read for the first time about Yamada's work on real-time computable functions [11,28]. These were, in essence, sets of integers for which a Turing machine on the nth operation printed a 1 if n was in the set and a zero otherwise. This could also be viewed as the characteristic function of the range of an increasing function. Yamada investigated properties of such functions in his dissertation and later showed that not all monotonic increasing computable functions can be so computed.

In retrospect it is surprising that Yamada or his thesis supervisor, Robert McNaughton, did not go on to develop a general theory for the classification of computable functions by their computation time requirements. They had very definitely made the first step and found an interesting family of easily computable functions.

Yamada's comment [28] on his work is interesting and quite revealing:

> "The present work, which is an attempt to construct a mathematical model for digital computers, concerns a class of multi-tape Turing machines with some constraints used to compute a certain class of functions in real time. In using digital computers, it is important to know the time required to compute a given function. However, to the best of this writer's knowledge, no attempt has ever been made to investigate a general theory of this sort. As a start in this direction, the writer has investigated the real-time computability of some recursive functions by means of a subclass of Turing machines. This investigation is by no means complete, and further research is required."

It is my personal impression that during this period many computer scientists hoped to find the "mathematical model for digital computers" which would model real computing. The unrestricted Turing machine was dismissed as an unrealistic model and one joked about the "Turing tarpit." There was a vague feeling that computations should be classified by their complexity, but both Yamada and Myhill, who defined and studied linearly bounded automata, followed the one model approach. I will return to Myhill's work in the discussion of tape bounded computations.

It is also strange to note that we did not follow up on Yamada's work right away. We outlined our book, read "Finite Automata and Their Decision Problems" by Rabin and Scott [17], studied regular expressions and obtained some initial results about recursively enumerable families of recursvie functions (which we were wise enough not to try to publish).

We also interviewed Mike Harrison. I usually recorded such visits in my Log Book and quite frequently they are followed by comments about the visitor; the juiciest comments are recorded in Latvian, which I consider as a reasonable secure code in computer science circles. Mike's visit, to my great puzzlement, produced neither English nor Latvian comments.

Only in early November did we start an intensive investigation of time bounded computations and realized that a rich theory about computational complexity could be developed. The first explicit mention of classifying functions by their computation time occurs in my Log Book on November 11 (which happened to be a Sunday). I believe that my thinking about these problems was very strongly motivated by a general belief that there must exist a quantitative theory of computing, that we must be able to measure the "computing work" done in computing and classify computations by their complexity. There is no doubt that my early background in physics influenced my thinking and that Shannon's Information Theory provided an example of a quantitative non-physical law. Finally, I believe that Yamada's work had a delayed effect on our first attempts to classify computation by their computation time.

The same week Stearns and I were deep in discussion about time limited computations, and we started referring to this topic as computational complexity. The weeks that followed were largely dedicated to an intensive study of time bounded computations and our progress was very rapid. About two weeks after Stearns and I initiated this investigation we knew that we had found an exciting research area and we started outlining a paper on computational complexity of algorithms.

The laconic entries from my G.E. Research Labratory Log Book, in which I recorded the main activities and events each day, probably give a good indication of our progress:

November 12, 1962: "Some work on "Comp. Complexity."
November 14: "Worked on T-comp. fu. with RES. Same nice insights."
November 28: "Work on Comp. Complex. This should be a good paper."
December 5: "Work on C. Complexity. More insights."
December 6: "More results on CC."
December 10: "Comp. Complexity starts to crystalize. Should be good paper."
December 12: "Writing of CC."

The Log Book ends with the entry:

December 31: "This was a good year."

At the end of 1962 and early 1963 we were on our way with a systematic investigation of computational complexity and we had obtained enough results to realize that we had found a very interesting and possibly an important new area of research. We believed that these were the beginnings of the quantitative theory of computation which we had been searching for. Furthermore, the results we were obtaining and the new problems and relations to other fields convinced us that a careful study of quantitative aspects of computing or constructive mathematics will enrich computer science and mathematics. Though, as you will see from the following, we were not completely sure who would be most interested in our new results.

By March of 1963 our manuscript "On the Computational Complexity of Algorithms" was finished and cleared by G.E. for outside publication. On March 15 it was submitted to the Journal of the ACM.

During the following weeks we wondered who would appreicate our work and where would it find the best reception. It seems that our mathematical training and indoctrination made us question our decision to submit our paper to an ACM journal. We resolved our doubts in a very unusual and possibly improper way. On March 28 we submitted our paper also to the Transaction of the American Mathematical Society.

The Transaction accepted our paper on July 31. They tried to convince us that mathematical journals should not have drawings of Turing machines in them, but after some discussion the drawings were left in. The paper was revised and expanded and appeared in the Transactions in May 1965 with nice pictures of Turing machines.

We apologized to ACM.

When I look back at this period I am struck by our independence and our very clearly articulated desire to help develop a quantititive theory of computing. We did not try to find the automaton which would model real computing or give elegant characterizations of subclasses of recursive functions. We simply wanted to measure the difficulty of computing or the "computing work" done. We had, if you permit the expression, an underlying philosophy and we, at least partially, were able to translate it into a mathematical theory.

We were surprisingly ignorant about effective computability when we started thinking about computational complexity, though otherwise we knew quite a bit of mathematics and our instincts lead us frequently to behave more like mathematicians than computer scientists. It may well be possible that our naivity and the nature of the field permitted us to pursue our desire to help construct a quantitative theory of computing more directly than if we had received a traditional instruction in related areas. In essence, we did not have a traditional framework in terms of which to study special classes of functions or to which to try to relate our results.

Somewhat facetiously one could say that by the time we heard about Grzegorczyk hierarchies and learned how to spell his name, we had developed our own ideas and were neither strongly influenced nor side tracked by Grzegorczyk's elegant work.

In this context it is interesting to recall what Myhill considered important in his study of linearly bounded automata [13]:

> "The principal result of this paper is that every rudimentary class of tapes is represented by some l.b.a. The term 'rudimentary' is due to Smullyan;..."

I believe that this is the reference which sent us looking for Smullyan's book, which we studied when we started learning about effective computability.

Similarly, J. Siegel responded to Yamada's work on real-time computations by showing that not every countable function is primitive recursive and not every monotonic increasing primitive recursive function is a real-time countable function [23].

Besides these results, we can find other examples in the literature, where preference is given to relating the new concepts and results to the old concepts and not immediately pursue the new ideas.

Our interaction with other people working on related problems was quite extensive and played an important role in our work. Even though at this time there were a lot fewer people working in computer science, it is impressive to see with how many we interacted and how quickly we found out what was happening in related research. Clearly, conferences played a considerable role and this symposium was particularly important in the following years.

In April of 1963 we visited Harvard where we met with Pat Fischer, Hao Wang and Michael Rabin and found out that Rabin had also worked on the complexity of computations. We exchanged ideas on complexity theory and discussed our work on time bounded computations. For us this was a memorable and very encouraging meeting. It is my impression that at this meeting we enticed Rabin to try to show that two-tape Turing machines are faster than one-tape machines, which eventually resulted in Rabins' paper "Real Time Computation" [16]. This is a very nice paper and it introduced new proof techniques. It furthermore showed that Rabin had mastered "Time" completely: the reprint states on the first page that the paper was published in the Israel Journal of Mathematics, Vol., No. 4, December 1963 and that the manuscript was "Received January 24, 1964."

After our meeting, we secured Rabin's technical report "Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets" and read it eagerly. The main result of this paper establishes under weak assumptions the existence of arbitrarily complex recursive zero-one functions. It is this report which led Manuel Blum to his elegant axiomatic definition of complexity measures. Only later, after reading Blum's paper, did we fully appreciate the value of the axiomatic approach to complexity theory and Rabin's insight.

We also found out that Bob Ritchie had worked on predictably computable functions [18] and studied his paper. It was probably from Ritchie's paper that we first heard about the Grzegorczyk hierarchy.

On October 7 and 8, 1963, Michael Arbib visited the G.E. Research Laboratory and Pat Fischer visited us on October 11. I mention these visits explicitly because one of these visitors left us a draft of Manuel Blum's work on axiomatic computational complexity, which later became his dissertation and was published as "A Machine-Independent Theory of the Complexity of Recursive Functions" in JACM in 1967 [2]. I have an old copy of Manuel's paper which has an inscription "To Juris Hartmannis ...Submitted to Trans.", and I am not clear why it appeared in JACM.

We were very much impressed by Blum's work, which in a way reformulated and extended Rabin's ideas, and proved some surprising results. In particular the speed-up theorem intrigueg us as it intrigued many others. I worked hard to understand the proof of this theorem and finally, on December 26, 1963, there is a Log Book entry:

> "I understand Blum's proof all the way. Nice proof!"

It was indeed a surprising result and a nice proof. We returned to this proof years later with John Hopcroft when we reworked it for our "An Overview of Computational Complexity Theory." I believe that currently Paul Young's proof is the most transparent one among the available proofs of this theorem.

During this time we also had the pleasure to meet Manual Blum personally at MIT and exchange ideas about our common interests in complexity theory. On the same trip we discussed the Krohn-Rhodes decomposition theory with Paul Zeiger, who wrote his Ph.D. dissertation on this topic and whose approach we followed in our book.

TAPE BOUNDED COMPUTATIONS

After finishing our paper on the complexity of computation time and the study of Blum's work we returned to finite automata, wrote our book and investigated several other research problems. We were visited by many people, some of whom we interviewed. Among the visitors were Dave Liu, Fred Hennie, Phil Dauber, John Hopcroft, Dan Younger, Michael Dertouzos, Pat Fischer and Paul Zeiger.

I lectured at University of Michigan, Harvard, Penn State; we visit MIT for discussions, and Dick Stearns and I each gave one of our two joint papers at the Fifth Annual Symposium on Switching Circuit Theory and Logical Design in October 1964 held at Princeton University. Stearns lectured "On the Application of Pair Algebra to Automata Theory" and I talked about "Computational Complexity of Recursive Sequences."

It is strange to look at the Forward of these proceedings and find there stated that:

> "From the many papers submitted in response to the committee's call for papers, these were selected on the basis of originality and relevance to problem of current interest in switching theory."

The program chairman of the 1963 Symposium put it even more directly:

> "Each was selected on the basis of originality and relevancy to switching theory. None of the manuscripts has been refereed..."

It may interest you to know that the chairman of the 1963 Symposium, quote above, was Seymour Ginsberg, who always has known how to express himself! Somewhat to my surprise, I discovered that the 1965 Symposium Forward was a verbatim copy of the 1964 Forward and that I was the chairman of the 1965 program committee.

Not quite believing that we had or could continue contributing papers of "originality and relevance to switching theory" we joined the irreverent and unsuccessful attempt at the Business

Meeting of the 1964 Symposium to change the name and hopefully the interests of this Symposium. As it is well known, we succeeded on the following try in 1965 to change the name of the Symposium and then again in 1975. I hope that we do not tinker with the name for the next twenty years.

In retrospect it is quite surprising that after our investigation of the complexity of time bounded computations and Blum's work on the axiomatically defined complexity measures nobody looked at tape or memory bounded complexity of computations for more than a year. This is particularly surprising since Bob Ritchie had used tape bounds to define the hierarchy of predicatably computable functions and Myhill had been quite explicit about memory bounded computations in his 1960 technical report on linear bounded automata [13]:

> "The measurement of memory capacity in bits appears to us to be heuristically valuable in the classification of generalized automata (though we have not been able to prove any theorems by this means). If by a total state of a machine we mean the internal state together with the state of its tape (or, more generally, tapes), then the memory capacity m of a machine can be taken as log s bits, where s is the number of total states. We may expect the power of a machine to increase as m increases."

It is interesting to look back at Myhill's "Linear Bounded Automata" report to find what triggered his study of these automata. I quote from [13]:

> "The immediate motivation for our present investigation is a short selection (II.2) occurring in the technical memorandum [3a] on which the Rabin-Scott paper was based: This section, of which no trace appears in the published version, concerns two-way automata with erasing, i.e. such that the reading had can change any of the ones or zeros on the tape to a new symbol 'blank', which cannot however subsequently be changed to a one or a zero. The principal result of this surpressed section is that these automata <u>can</u> accomplish more than classical finite automata..."

The restriction to only erasing tape symbols looked artificial to Myhill, as it must have to Rabin and Scott who removed this section from the published version of their paper. Myhill went on to define the linearly bounded automaton as a more natural model for computers. Later the lba gained in importance as its relation to context-sensitive languages was discovered and when it raised fundamental questions about non-deterministic computations. Clearly, this model has played an important role in automata theory and it is interesting to keep in mind what initiated its investigation. I believe this is a very good example of how an innocuous and unnatural model can trigger a fruitful investigation.

On the lighter side, it is interesting to note that Myhill refers to the Rabin-Scott paper in [13] as "Finite Automata And Their Problems." Yes, John, we always knew that finite automata had problems.

In spite of all the hints about the possibility to investigate bounded memory computations we did not look at this topic until late in 1964.

Looking back at these events, I have the feeling that we (and possibly others) did not consider sufficiently seriously the developments in the whole field and try to assess what was really worthwhile doing as well as what had changed because of recent events. I still have the feeling that most of us, and certainly I, do too little introspection about the developments in our research area and miss important ideas because of misplaced enthusiasm for our own current research problems and results.

Very late in 1964 we suddenly turned with Phil Lewis and Dick Stearns to the study of tape bounded computations. I have no idea what triggered this sudden interest, but I suspect that Phil Lewis may have provided the first impetus by asking the right questions.

My Log Book entry for December 22, 1964 reads:

> "With RLS, PML discovered that we can classify by amount of tape needed languages, etc."

My G.E. Information Studies Project Report Report for December 1964 states it more explicitly:

> "The most important event of this period was the discovery with P. M. Lewis and R. E. Stearns that abstract languages can be classified by the amount of tape which is needed in an automaton to recognize them. In order to do this, we defined several new automata models and derived quite a few results about them. I believe that this opens up a new and interesting area of work."

Again, once we started our investigation our progress was quite rapid and with pleasure we noticed that the tape bounded computations yielded several sharper results than we could obtain for time bounded computations.

By March of 1965 we had finished a report on this work, which we presented at the IFIP Congress of 1965 [6]. Two papers at the 1965 Symposium on Switching Circuit Theory and Logical Design gave a more detailed exposition of this work [10,26]. This time the name of the Symposium was changed!

In this case, it is also informative to look back at what we saw but did not fully understand. We proved that there was a gap in the complexity classes from tape bound

$$L(n) = c \text{ to } L(n) = \log \log n.$$

We found this very strange, but never suspected that gaps appear in all complexity measures and for artitrarily complex computations. This was discovered independently by Trakhtenbrot [27] and somewhat later by Borodin [1] in his Ph.D. dissertation. The Gap Theorem and the work which it initiated added considerably to our understanding of computational complexity.

We also showed that context-free languages could be recognized on

$$L(n) = [\log n]^2$$

tape with a proof that involved the "recursive guessing of the middle of the computation." This permitted us to use only log n instantaneous descriptions of length log n to simulate deterministically all possible exponentially long computation. Quite surprisingly, I personally appreciate the full power of this method only after I read Savitch's paper [19] with the elegant

$$NDTAPE[L(n)] \subseteq DTAPE[L(n)^2]$$

result, which used this method. I personally never suspected that the above gap could be less than exponential.

In the fall of 1965 I joined the newly formed Computer Science Department at Cornell University and a new phase started in my association with computer science research and education.

EPILOGUE

As I look back at the twenty years of FOCS and more generally at the developments in theoretical computer science during this period, I am struck by two almost contradictory impressions. On the one hand, I am deeply impressed by how well we have done. The progress in a number of problem areas in theoretical computer science has been far more rapid than we could have expected and many concepts and results have yielded deep insights. Furthermore, some parts of our field have been unified and structured in a beautiful way and some unexpected and interesting connections between problems and other research areas have been revealed. The researchs have in general shown flexibility and searched out new areas and developed new techniques and theories.

On the other hand, I am also struck by how much of what we have done has been of no lasting value. The field is littered with innumerably papers of dubious quality, with papers which hide their shallowness (probably also from the authors) behind obscure mathematical formalizations and deal with minute problems of no overall importance.

At the same time, we have to understand that much of the thinking and research in this area is just the clearing out of the "intellectual underbrush," the testing and exploring of ideas and models until we see where the important ideas, problems and results are hidden. Our research and publications are furthermore a communal learning and educational process which shapes future research. It is a complicated process of probing the unknown, trying to grasp the right concepts, to formulate the right models and gain the necessary results and insights.

As a matter of fact, since computer science deals with man-made objects and therefore is not a physical science, we do not have an existing universe to explore and explain. We cannot proceed like many other better established sciences have and depend completely on their paradigms and methodology. In computer science we must first imagine or build what we want to study and then develop the concepts, formalizations and theories to be able to think about the possibly systems and solutions and pick a good one from them. We must develop the intellectual tools not only to explore the existing but to study the possible, to help us imagine it, to build and analyse it or analyse and build it.

Therefore I believe that research in computer science will require extensive exploratory theorizing until we isolate the right conceptualizations and discard many others which do not capture the reality we want to understand.

The high standards and norms of scholarship of the older sciences have served computer science very well by guiding its early developments. At the same time, we have also blundered extensively by either assuming that computer science must behave very much like the older experimental sciences or that it must follow the norms of pure mathematics.

In particular in theoretical computer science we have been guilty of behaving too much like pure mathematicians; the mathematicians compass has not always guided us well in exploring computer science. Time and again we have valued the difficulty of proofs over the insights the proven results give us about computing, we have been hypnotized by mathematical elegance and pursued abstraction for its own sake. Frequently we have practiced "intellectual counterpunching" by staying with small, previously defined (and possibly) irrelevant problems instead of searching for new formulations and development of theories more directly related to computing.

Still, everything considered, I am optimistic that we will develop a deeper understanding of the nature of computer science and forge new relations between experimental and theoretical computer science which will influence and further its future developments. I have no doubts that computer science can mature to a deep and influential science. I expect that there will be some profound and surprising insights. I am convinced that we will discover, just like we found in other sciences as they matured, that the laws which we will discover about information processing do not feel any obligation to conform to our current naive ideas

derived from very limited experience with small machines and simple problems (or our dim perception how animals process information). I believe that as we explore information processing further there will be startling surprises and that our current ideas about computing will have to be modified substantially.

Through fortunate circumstances or wise choices, the participants of this Symposium have been permitted to participate and sometimes even influence the development of a new science. A new science which is still in its infancy but whose depth, importance and future possibilities can be perceived. I hope that we are wise and clever enough to understand its nature, not to underestimate its intellectual richness and depth, and that we will contribute to some of its major developments during the next twenty years.

[1] A. Borodin, "Computational Complexity and the Existence of Complexity Gaps," JACM Vol. 19 (1972), p. 158-174.

[2] M. Blum, "A Machine-Independent Theory of the Complexity of Recursive Functions," JACM Vol. 14 (1967), p. 322-336.

[3] N. Chomsky "Context-free Grammer and Pushdown Storage," Quarterly Progress Report No. 65, Res. Lab. of Electronics, M.I.T., 1962.

[4] M. Davis, "Computability and Unsolvability," McGraw Hill, New York, 1958.

[5] J. Hartmanis, "On the State Assignment Problem for Sequential Machines. I," IRE Trans. on Electronic Computers, Vol. 10 (1967), p. 157-165.

[6] J. Hartmanis, P.M. Lewis II, and R.E. Stearns, "Classification of Computations by Time and Memory Requirements," Proceedings of IFIP Congress 65, Vol. 1 (May 1965), p. 31-35.

[7] J. Hartmanis and R.E. Stearns, "Algebraic Structure Theory of Sequential Machines," Prentice-Hall, Inc., Englewood Cliffs, NJ, 1966.

[8] H. Hermes, "Aufzaehlbarkeit, Entscheidbarkeit, Berechenbarkeit," Springer-Verlag, 1961.

[9] K.B. Krohn and J.L. Rhodes, "Algebraic Theory of Machines," Proceedings of the Symposium on Mathematical Theory of Automata, April 1962, Microware Research Institute Symposium Series, Vol. XII, Polytechnic Press of the Polytechnic, Institute of Brooklyn, NY, 1963, p. 341-384.

[10] P.M. Lewis II, R.E. Stearns, and J. Hartmanis, "Memory Bounds for Recognition of Context-Free and Context-Sensitive Languages," 1965 IEEE Conference Record on Switching Circuit Theory and Logical Design (October 1965), p. 191-202.

[11] R. McNaughton, "The Theory of Automata - A Survey," Advances in Computer, F.L. Alt, editor, Vol. 2, Academic Press, NY and London, 1961.

[12] E.F. Moore, "Gedanken-Experiemtns on Sequential Machines," Automata Studies, Annals of Mathematical Studies, Vol. 34, eds. C.E. Shannon and J. McCarthy, Princeton University Press, 1956, p. 123-153.

[13] J. Myhill, "Linear Bounded Automata," University of Pennsylvania, Report No. 60-22, June 1960.

[14] J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Automata Studies, Annals of Mathematical Studies No. 34, eds. C.E. Shannon and J. McCarthy, Princeton University Pres, 1956, p. 43-98.

[15] M.O. Rabin, "Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets," Technical Report No. 2, Hebrew University, Jerusalem, Israel, 1960.

[16] M.O. Rabin, "Real Time Computation," Israel J. of Mathematics, Vol. 1 (1963), p. 203-211.

[17] M.O. Rabin and D. Scott, "Finite Automata and their Decision Problems," IBM J. Res. Vol. 3 (1959), p. 115-125.

[18] R.W. Ritchie, "Classes of Predictably Computable Functions," Trans. Amer. Math. Soc. (1963), p. 139-173.

[19] W.J. Savitch, "Relations Between Nondeterministic and Deterministic Tape Complexities," J. Computer and Systems Sciences, Vol. 4 (1970), p. 177-192.

[20] C.E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," Trans. AIEE Vol. 57 (1938), p. 713-723.

[21] C.E. Shannon, "Mathematical Theory of Communication," Bell Syst. Techn. J., Vol. 27 (1948), p. 379-423; 623-658.

[22] C.E. Shannon and J. McCarthy, Editors, "Automata Studies," Annals of Math. Studies, No. 34, Princeton University Press, Princeton, NJ, 1956.

[23] J. Siegel, "Some Theorems about Yamada's Restricted Class of Recursive Functions," IBM Research Paper RC-510, IBM T.J. Watson Research Center, 1961.

[24] R.M. Smullyan, "Theory of Formal Systems," Annals of Math. Studies, No. 47, Princeton University Press, Princeton, NJ, 1961.

[25] R.E. Stearns and J. Hartmanis, "On the State Assignment Problem for Sequential Machines. II," IRE Trans. in Electronic Computers, Vol. 10 (1961), p. 593-603.

[26] R.E. Stearns, J. Hartmanis and P.M. Lewis II, "Hierarchies of Memory Limited Computations," 1965 IEEE Conference Record on Switching Circuit Theory and Logical Design," (October 1965), p. 179-190.

[27] B.A. Trakhtenbrot, "Turing Computations with Logarithmic Delay," Algebra i Logika, Vol. 3 (1964), p. 33-48.

[28] H. Yamada, "Real-Time Computation and Recursive Functions not Real-Time Computable," General Dynamics Corporation, Rochester, NY, July 1960, Revised November 1961.

[29] H. Yamada, "Real-Time Computation and Recursive Functions not Real-Time Computable," IRE Trans. on Electronic Computers, Vol. EC11 (1962), p. 753-760.