The Origins of Computational Complexity

Some influences of Manuel Blum, Alan Cobham, Juris Hartmanis and Michael Rabin

Abstract. This paper discusses some of the origins of Computational Complexity. It is shown that they can be traced back to the late 1950's and early 1960's. In particular, it is investigated to what extent Manuel Blum, Alan Cobham, Juris Hartmanis, and Michael Rabin contributed each in their own way to the existence of the theory of computational complexity that we know today.

1. Introduction

The theory of computational complexity is concerned with the quantitative aspects of computation. An important branch of research in this area focused on measuring the difficulty of computing functions. When studying the computational difficulty of computable functions, abstract models of computation were used to carry out algorithms. The goal of computations on these abstract models was to provide a reasonable approximation of the results that a real computer would produce for a certain computation. These abstract models of computations were usually called "machine models".

In the late 1950's and early 1960's, the study of machine models was revived: the blossoming world of practical computing triggered the introduction of many new models and existing models that were previously thought of as computationally equivalent became different with respect to new insights regarding time and space complexity measures.

In order to define the difficulty of functions, computational complexity measures, that were defined for all possible computations and that assigned a complexity to each terminating computation, were proposed. Depending on the specific model that was used to carry out computations, there were many notions of computational complexity. As Peter van Emde Boas, a mathematical computer scientist, put it in an abstract of one of his articles [8, ch.1]: "*We must cope with the discrepancy between a machine based foundation of complexity notions and a machine independent intuition on what computational complexity means*" . But the existence of many different machine models did not necessarily imply that there were no **uniform underlying notions of computational complexity.**

2. Alan Cobham

To comply to these uniform notions, a theorem on machine independent computational complexity will have to be true for every machine model and measure of computational complexity. I believe the mathematician Alan B. Cobham was aware of this when he wrote his paper `The intrinsic computational difficulty of functions' in 1964 [14][3]. Cobham posed two questions at the beginning of his paper: first, is it harder to multiply than to add? and second, why? He felt the answer to the first question should surely be yes, but to be able to prove it and answer the second question he would have to show that there is no

algorithm for multiplication *computationally* as simple as that for addition, "*and this provides something of a stumbling block"* [3, p.24]. He recognized the need for a single theory that is true for every machine model :

> [The first question] is concerned with the computational difficulty of these two operations [multiplication and addition] without reference to specific computational methods; in other words, with properties intrinsic to the functions themselves and not with properties of particular related algorithms. [3, p. 24]

He also pointed out that he did not commit himself to any complexity measure in particular:

> Although I have so far been tacitly equating computational difficulty with time and storage requirements, I don't mean to commit myself to either of these measures. It may turn out that some measure related to the physical notion of work will lead to the most satisfactory analysis: or we may ultimately find that no single measure adequately reflects our intuitive concept of difficulty. [3, p. 27]

He did not answer his two questions that were mentioned above; he believed that they could only be answered once the theory (of computational complexity) was properly developed. His paper contributed to this development by emphasizing two very important questions: what is the right complexity measure (if it exists) and, what constitutes a "step" in a computation? Defining a step in a computation is closely related to determining the best computer model for measuring the computation time of an algorithm.

Another contribution of Cobham's paper to the field of computational complexity was the definition and characterization of the class *L* of functions which are computed by Turing machines in a number of steps bounded by a polynomial in the length of the input. Cobham pointed out that although he involved Turing machines in his theorem, the theorem remains correct even if one considers far wider classes of computing machines. That is, the set of problems computable in polynomial time remains independent of the particular deterministic model.

3. Juris Hartmanis

Cobham was not the only one in the early 1960's who concerned himself with the question of intrinsic computational difficulty. In 1965 the paper `On the computational complexity of algorithms' written by Juris Hartmanis in collaboration with Richard E. Stearns was published (the manuscript of this paper was finished in 1963 [7, p.47]). In the introduction, Hartmanis states:

> One finds, however, that some computable sequences are very easy to compute whereas other computable sequences seem to have an inherent complexity that makes them difficult to compute. In this paper, we investigate a scheme of classifying sequences according to how hard they are to compute. [6, p.285]

At least one of the influences of this paper on the field of computational complexity is still very much noticeable today: it is very likely here that the term "computational complexity" was introduced for the first time in a peer-reviewed article [16, p.6][4, p.401]:

> … this scheme can be generalized to classify numbers, functions, or recognition problems according to their computational complexity. [6, p.285]

Hartmanis measured the computational complexity of a sequence by how fast a multitape Turing machine can print out the terms of the sequences. At that time, much of the work in this area was stimulated by the rapidly growing importance of computation through the use of digital computers . This influenced Hartmanis to choose a multitape Turing machine as an abstract model of a computing device be-

cause he was convinced that all digital computers in a slightly idealized form belong to the class of multi-tape Turing machines [6, p.285]. In their paper, Hartmanis and Stearns also investigated how changing the choice of abstract machine model might affect the computational complexity. They showed that some abstract machine models can simulate other abstract machine models by at most squaring the computation time.

Later, in 1981 Hartmanis wrote in an historical account of this period that when he and Stearns started thinking about computational complexity, they were surprisingly ignorant about effective computability, and because they knew quite a bit of mathematics their instincts led them frequently to behave more like mathematicians than computer scientists [7, p.47]. In essence, they did not have a traditional framework in terms of which to study special classes of functions or to which to try to relate their results. They did not try to find "the" automaton that would model real computing or give elegant characterizations of subclasses of recursive functions. They simply wanted to measure the difficulty of computing. Hartmanis also explains in this historical account that it was only after `On the computational complexity of algorithms' was written that he met Rabin and exchanged ideas with him on complexity theory. After meeting with Rabin Hartmanis studied Rabin's work [11], but he admits that it was only later, after reading Blum's paper [1], that he did fully appreciate the value of the axiomatic approach to complexity theory and Rabin's influence [7, p.47].

Hartmanis contributed to the theory of computational complexity with a very mathematical inclined approach. As we shall see next, the contributions of Michael Rabin and Manuel Blum were more influenced by logic.

4. Michael Rabin

Around 1960, Michael Rabin was probably one of the first to address the general question of the relative computational difficulty of computable functions explicitly [4, p.401][5, p.473] in his papers `Speed of computation of functions and classification of recursive sets' and `Degrees of difficulty of computing a function and a partial ordering of recursive sets' [10][11] . He wondered what it meant to say that `a function f is more difficult to compute than a function g'. Rabin suggested an axiomatic framework that provided the basis for the abstract complexity theory developed by Blum and others (see Section 5).

In 1959 Rabin published his paper `Finite automata and their decision problems' [12] that he had written together with the logician Dana Scott. In this paper, Rabin noted that although Turing machines were widely considered to be the abstract prototype of digital computers, to him it seemed as if workers in the field felt more and more that the notion of a Turing machine was too general to serve as an accurate model of actual computers. Rabin attributed this to the fact that even for simple calculations it is impossible to give an a priori upper bound on the amount of tape needed by a Turing machine to carry out the calculation at hand [12, p. 114].

Rabin engaged with the idea of a finite automaton that had appeared in the literature at that time. These finite automata were machines that had only a finite number of internal states that could be used for memory and computation. To Rabin it seemed that this restriction of finiteness allowed finite state automata to give a better approximation to the idea of a physical machine than Turing machines could give. He defined automata that were closer to the idea of a Turing machine than the existing finite au-

tomata at the time, while preserving the important feature of using only a pre assigned amount of tape. He was aware that finite state machines could not do as much as a Turing machine but he did not see this as a limitation for practical applications [12, p. 114]. Rabin used these automata in defining an intrinsic mathematical formalization of definable sets (Rabin and Scott continued on work from Nerode, Myhill, and Shepherdson, see Section 6).

5. Manuel Blum

In 1967, just after Hartmanis and Stearns had written their paper on the computational complexity of algorithms, Manuel Blum, in his Ph.D. dissertation at MIT titled `A machine-independent theory of the complexity of recursive functions' [1], developed an axiomatic theory of computational complexity and, among many other results, showed that all complexity measures are recursively related. The speed-up results from Hartmanis and Stearns were special cases of this relationship [6].

Blum did his work on 'A machine-independent theory of the complexity of recursive functions' as a student at MIT after attending lectures by Rabin and reading Rabin's unpublished report [11] [16, p.8]. Blum pointed out in his work [1, p.322] that the theory he developed expanded along lines suggested by Rabin's axiomatic approach [11]. In his paper, Blum introduced his theory as follows:

> … so all hope that an individual function might enjoy a unique measure of complexity, one that is independent of the class of machines, must vanish. What remains possible, and is offered here instead: an axiomatic theory whose *theorems* are independent of class.

Blum presented two axioms that any reasonable complexity measure should satisfy and then described some basic properties of any such complexity measure. Informally stated, the first axiom said that any way of measuring complexity should assign a value to those (and only those) computations which stop. The second axiom said that it should always be decidable if a computation has a specified measured value. The surprising fact is that these two axioms were sufficient to prove many interesting results about all complexity measures for which they hold.

In `On the size of machines' [2], Blum presented an axiomatic approach to measuring the length of a program or the size of a machine computing a recursive function. His approach was very similar to his approach to measuring the length of a computation discussed above. The method considered properties of recursive functions and did not consider particular properties of any class of machines.

6. Closing Remarks

How did the work of Blum, Cobham, Hartmanis, and Rabin in the late 1950's and early 1960's influence the existence of the field of computational complexity as we know it today? All four authors were interested in the quantitative aspects of computing and approached this topic from their own point of view.

In his research of the quantitative aspects of computing, Hartmanis followed a more mathematical approach than Rabin and Blum, whose work was more intertwined with logic. Both Rabin and Blum were trying to develop an axiomatic framework for a machine independent theory and their work can be viewed as machine independent from the outset. The work of Hartmanis was based in a machine dependent setting but still allowed derivation of machine independent results. Cobham was also aware of

the need for a machine independent theory to be able to measure the intrinsic computational difficulty of functions.

Many results of the work of Blum, Cobham, Hartmanis still influence work in the field of computational complexity that is done today. Just a few examples are: the class P of functions solvable in polynomial time defined by Cobham, the speed-up theorem developed by Blum, the concept of nondeterministic machines introduced by Rabin, and the basic concept of a `computational complexity class' proposed by Hartmanis. These results (among much more work) provide a solid argument for the visible contribution to early computational complexity theory by the aforementioned authors. I am therefore tempted to jump to the conclusion that the field of computational complexity today would not be the same without these contributions. But this would be a wild conclusion that I could not possibly defend as there were many other researchers in the late 1950's and early 1960's who made major contributions that were not discussed in this paper (for example, the introduction of the notion of real-time computability around 1960 by the Japanese computer scientist Hisao Yamada [17]). Therefore, it does not seem unreasonable to assume that if the contributions of Blum, Cobham, Hartmanis and Rabin hadn't occurred, other researchers would eventually have contributed similar results.

Blum, Cobham, Hartmanis and Rabin were however among the first to deal with the quantitative aspects of computing. For various reasons their published work found its way to many people and inspired the work of numerous researchers. This is easily seen by the vast amount of papers that are published today which still refer to the work of Blum et al. I therefore conclude that without the work of Rabin, Blum, Cobham and Hartmanis the field of computational complexity probably would not be as far progressed as it is today.

The reader should be aware that this work is by no means intended to be a complete overview of all historic influences on the foundations of the modern field of computational complexity. Many people that influenced this field have been left out of this discussion.

References

[1] M. Blum. A machine-independent theory of the complexity of recursive functions. *J. ACM 14*, 322- 336, April 1967.

[2] M. Blum. On the size of machines. *Inf. Contr. 11*, 257-265, 1967.

[3] A. Cobham. The intrinsic computational complexity of functions. *Proc. 1964 Int. Congress on Logic, Methodology and Philosophy of Science*, North-Holland, Amsterdam, 24-30, 1965.

[4] S. A. Cook. An overview of computational complexity. *Communications of ACM, Vol. 26, No. 6*, 401-402, June 1983.

[5] J. Hartmanis, and J. Hopcroft. An overview of the theory of computational complexity. *J. ACM 18*, 444-475, 1971.

[6] J. Hartmanis, and R. E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. 117*, 285-306, May 1965.

[7] J. Hartmanis. Observations about the development of theoretical computer science. *Annals of the history of computing, Vol. 3, No. 1*, January 1981.

[8] J. van Leeuwen (ed.). Handbook of theoretical computer science (2nd Vol.). Amsterdam; New York: Elsevier; Cambridge, Mass.: MIT Press, 1990.

[9] M. S. Mahoney. Computer Science: The search for a mathematical theory. *Science in the 20th Century*, ed. J. Krige and D. Pestre, Chap. 31, 1997.

[10] M.O. Rabin. Speed of computation of functions and classification of recursive sets. *Proc. third conven. of scientific societies,* Israel, 1-2, 1959.

[11] M. O. Rabin. Degrees of difficulty of computing a function and a partial ordering of recursive sets. Tech. rep. 2, Hebrew U. Jerusalem, Israel, April 1960.

[12] M. O. Rabin, and D. Scott. Finite automata and their decision problems. *IBM J. Res. 3*, 115-25, 1959.

[13] R. W. Ritchie. Classes of predictably computable functions. *Trans. Amer. Math. Soc. 106*, 139-173, 1963.

[14] J. Shallit. Blog entry titled "Alan Cobham". Written: March 31, 2010 Retrieved: June 18, 2014 from: http://www.recursed.blogspot.nl/2010/04/alan-cobham.html

[15] D. Shasha. An interview with Michael Rabin. *Com. of the ACM, vol. 53*, No. 2, February 2010.

[16] R. E. Stearns. Juris Hartmanis: the beginnings of computational complexity. In: *Complexity theory retrospective*, ed. A. L. Selman, Springer- Verlag, Chap 2, 1990.

[17] H. Yamada. Real-time computation and recursive functions not real-time computable. *Rochester, N.Y., General dynamics Corp.,* July 1960, revised November 1961.