Devin Connolly
15315916

The Work and Impact of Robert W Floyd

A Brief Biography

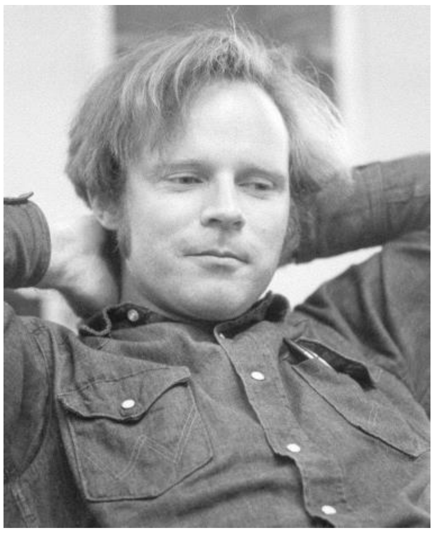
Robert W (Bob) Floyd (June 8, 1936 – September 25, 2001) was born in New York City. He received a Bachelor's degree in liberal arts in 1953 (at aged 17) from the University of Chicago, and a second Bachelor's degree in physics in 1958. By the time he was 27, Floyd was an associate professor at Carnegie Mellon University, and within 6 years he was a full professor at Stanford University.

Floyd's introduction to computing came from an early job as a computer operator at the Armour Research Foundation of the Illinois Institute of Technology. His curiosity led him to become a programmer by reading manuals, and at the same time he started his research career by publishing a paper on radio interference. At armour he became interested in the compilers that translate high level languages into machine ode. He soon published papers on both new notation for symbol manipulation systems and a new method of scanning arithmetic expressions (*'An Algorithm for coding efficient arithmetic operations')* (1).

In 1962 Floyd became a Senior Project Scientist at Computer Associates. He worked on compilers and published additional papers in the area. In 1966, Donald Knuth was preparing for the chapter of his series *'The Art of Computer Programming'* dealing with compilers and syntax analysis, and he noticed that "only five really good papers about compilers had been written so far, and Bob had been the author of all five" (2). Bob Floyd is the most cited author in Knuth's *The Art* series, and was the main proof reader and critic of the series.

Along with his research work, Floyd also invented many important practical algorithms. Best known are those that find the shortest paths through networks, compute the median of data, and render grey-scale images with binary pixels using error diffusion – the *Floyd-Steinberg* algorithm.

Regarding his personal life, Floyd had a strong social conscience and was a leading member of Amnesty International. He loved hiking and rock climbing and was an avid backgammon player. Floyd had his middle name changed to the single letter "W", but he often wrote it as an abbreviation with a period. He was married and divorced twice, and had four children.



Pictured is Bob Floyd. To simulate grey-scale with binary pixels, this photo has been rendered here using the Floyd–Steinberg "error diffusion" algorithm, implementing that algorithm exactly as suggested in the famous article that Floyd published with Louis Steinberg in 1976 (diffusing errors from bottom to top and right to left); the resolution is 600 dots per inch (4).

The Work of Bob Floyd

Floyd received a number of honours and awards over the course of his life, including the ACM Turing Award in 1978. The citation for his Turing award is as follows:

"For having a clear influence on methodologies for the creation of efficient and reliable software, and for helping to found the following important subfields of computer science: the theory of parsing, the semantics of programming languages, automatic program verification, automatic program synthesis, and analysis of algorithms."

As well as the aforementioned '*An Algorithm for coding efficient arithmetic operations'*, Floyd published many other successful and influential works. In 1967, Floyd built on earlier work of Alan Perlis, Saul Gorn and John McCarthy for proving programs correct. He developed a notation, initially for flowcharts and later for real programs, that assigned conditions at teach branch and entry point in the program. Errors were often found in large programs years after they had been put into production, and Floyd looked to remedy this. Some of these conditions related to the value of variables, and ensured that if these conditions were true upon entry they could be proven true at exit. Other conditions proved a program would halt, by requiring that, at each step, some value would decrease that could not decrease indefinitely (3).

'*Assigning meanings to programs*' was published in 1967 and represented the beginning of the theory of programming language semantics, and was a radical break from the past. Programmers had previously had no way to verify the correctness of their work, except by trial and error; in fact, programming was considered to be an inherently different sort of human activity, having no relation to mathematical rigor, so software engineers didn't even realize that there lacked any way to prove that programs were correct. A programmer's task was simply to fiddle with programs until no more mistakes could be found, and to hope that nothing had been overlooked. Floyd's paper introduced so-called "invariant assertions," which opened the eyes of the computer science world and led directly to a long series of future developments (2). '*Assigning Meanings to Programs'* was influential and inspired Tony Hoare to develop a system known as Hoare Triples that furthered his work (2).

Right: An excerpt from this paper, detailing the notation Floyd used when dealing with flowcharts, which he then expanded to apply to actual programs.

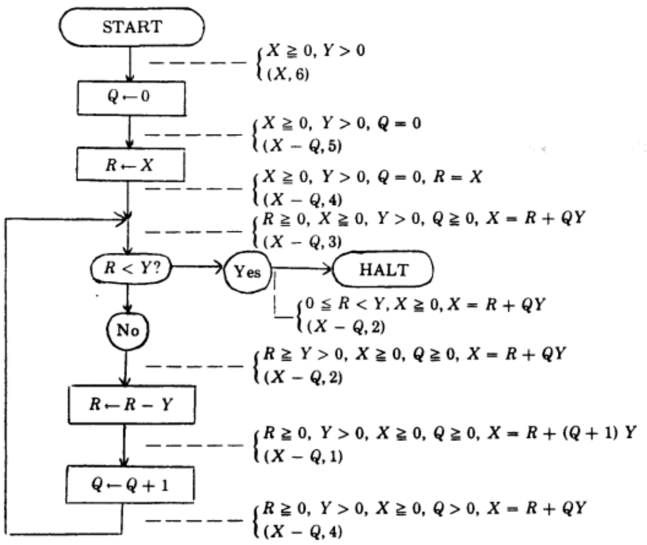

START
X ≧ 0, Y > 0 (X, 6)
Q ← 0
X ≧ 0, Y > 0, Q = 0 (X − Q, 5)
R ← X
X ≧ 0, Y > 0, Q = 0, R = X (X − Q, 4)
R ≧ 0, X ≧ 0, Y > 0, Q ≧ 0, X = R + QY (X − Q, 3)
R < Y?
Yes
HALT
0 ≤ R < Y, X ≧ 0, X = R + QY (X − Q, 2)
No
R ≧ Y > 0, X ≧ 0, Q ≧ 0, X = R + QY (X − Q, 2)
R ← R − Y
R ≧ 0, Y > 0, X ≧ 0, Q ≧ 0, X = R + (Q + 1) Y (X − Q, 1)
Q ← Q + 1
R ≧ 0, Y > 0, X ≧ 0, Q > 0, X = R + QY (X − Q, 4)

FIGURE 5. Algorithm to compute quotient Q and remainder R of $X \div Y$, for integers $X \geqq 0$, $Y > 0$

The Impact of Bob Floyd

"Floyd's 1960s method of invariants, in which assertions are attached to points in a computer program, is still the basis of much work in proving that computer programs meet their specifications," says John McCarthy, professor emeritus of computer science (5). As well as this, Floyd may have been the first advocate of refactoring -- the rewriting of working programs from scratch, re-using only the essential ideas. Refactoring is now standard practice among computer programmers. By continuously looking for simpler ways to do the same thing, Floyd aimed to improve not only programs but also programmers' abilities and understanding.

Hoare logic (also known as Floyd–Hoare logic or Hoare rules) is a formal system with a set of logical rules for reasoning rigorously about the correctness of computer programs. It was proposed in 1969 by the British computer scientist and logician Tony Hoare, and subsequently refined by Hoare and other researchers. The original ideas were seeded by the work of Floyd and his system for flowcharts discussed previously. The central feature of Hoare logic is the Hoare triple which describes how the execution of a piece of code changes the state of the computation (6).

Floyd's work changed the direction of development of programming languages, specifically regarding semantics. In his Turing Award lecture, *'The Paradigms of Programming',* he stated:

*"To the designer of programming languages, I say: unless you can support the paradigms I use when I program, or at least support my extending your language into one that does support my programming methods, I don't need your shiny new languages. [...] To persuade me of the merit of your language, you must show me how to construct programs in it." (*7)

In 1991, the Institute of Electrical and Electronics Engineers (IEEE) Computer Society awarded Floyd its Computer Pioneer Award for his work on early compilers (a compiler is software that translates a computer program as a whole into machine code that is saved for subsequent execution at a desired time). He made discoveries that revolutionized the field; for example, his paper of 1963, "*Syntactic analysis and operator precedence,*" represented the birth of practical methods to derive language translators from specifications of grammar. His seminal paper of 1964, "*The syntax of programming languages -- A survey,*" (8) not only brought order out of chaos by elegantly summarizing all previous work on compiler parsing but also introduced a significant new paradigm that led to the all-important notion of "object-oriented languages." (5)

Conclusion

Robert W Floyd changed the way we program, and changed the way programming as a whole is conducted, from error checking to parsing to specific algorithms themselves. He was presented with the Turing Award for his contributions to modern computer science, and was no doubt deserved of this accreditation.

Bibliography

(1) Floyd, RW, 1961. ‘An algorithm for coding efficient arithmetic operations’. *Communications of the ACM*, Volume 4 Issue 1, 42-51.

(2) Amturing.acm.org. (2017). *Robert W. Floyd Additional Materials - A.M. Turing Award Winner*. [online] Available at: http://amturing.acm.org/info/floyd_3720707.cfm [Accessed 8 Nov. 2017].

(3) Floyd, RW, 1967. ‘Assigning Meanings to Programs’. *Proceedings of Symposia in Applied Mathematics, Mathematical Aspects of Computer Science,* Volume 19.

(4) Knuth, D. (2017). *Robert W Floyd, In Memoriam*. [online] Amturing.acm.org. Available at: http://amturing.acm.org/p3-knuth.pdf [Accessed 8 Nov. 2017].

(5) Levy, D. (2001). *Robert Floyd, pioneer in computer programming, dead at 65. [online]* News.stanford.edu. Available at: https://news.stanford.edu/news/2001/november7/floydobit-117.html [Accessed 8 Nov. 2017].

(6) Stanford CS. (2008). *Hoare Logic* [online] Cs.stanford.edu. Available at: https://cs.stanford.edu/people/eroberts/courses/soco/projects/2008-09/tony-hoare/logic.html [Accessed 8 Nov. 2017].

(7) Floyd, RW, 1979. ‘Turing Award Lecture 1978’, *Communications of the ACM* Volume 22 Issue 8, 455–460.

(8) Floyd, RW, 1964. ‘The Syntax of Programming Languages – A survey’ Available at: http://www.win.tue.nl/~mvdbrand/courses/seminar/0809/papers/Floyd.pdf [[Accessed 8 Nov. 2017].