Published in *Bulletin of the European Association for Theoretical Computer Science*, Number 36, October 1989, guest column in the 'Algebraic Specification Column,' pages 96–102; also, in *Current Trends in Theoretical Computer Science: Essays and Tutorials*, World Scientific, 1993, pages 76–81.

Memories of ADJ

Joseph A. Goguen
Programming Research Group, University of Oxford
SRI International, Menlo Park CA 94025

In this guest column, I wish to present some informal and personal recollections of the exciting early days of ADJ. This time was very important to me personally, and I think was also significant for the field of algebraic specification. Because these recollections are so personal, they are also necessarily biased, and probably inaccurate in some respects; but perhaps this will just make the result more amusing to read.

In 1969, I took up a post in the Committee on Information Sciences at the University of Chicago, and began teaching things like the theories of information, automata and formal languages, and algebraic foundations of computer science. I also attended Professor Saunders Mac Lane's lectures and seminars on category theory. In 1970, Mac Lane suggested that I apply for the 1971-72 Research Fellowship in the Mathematical Sciences, which had just been established by the Mathematical Sciences Department at IBM's T.J. Watson Research Center (in Yorktown Heights, NY).

I was eager to go, because I wanted to deepen my understanding of theoretical computer science by working with this group whose research I already knew and respected, including Cal Elgot, Jim Thatcher, Eric Wagner and Jesse Wright. IBM's literature said that part of the purpose of this particular Fellowship was to expose the Mathematical Sciences Department to new research ideas. I took this very seriously, because I felt there were many interesting connections between category theory and computer science, especially in the theories of automata and formal languages. I felt that category theory should be especially relevant, because it could help with formalization, which is a major activity in theoretical computer science, and can be quite difficult. Finally, it seemed a pity that four people who worked in the same department and shared such an (at that time) esoteric interest as applying category theory to computer science, had never set up a joint working group on this topic.

Thatcher, Wagner and Wright became interested in such a group. One thing that I think especially caught their interest was the simple proof of the associativity of term substitution that I was able to show them. This proof used the freeness of term algebras to chase one simple diagram, in striking contrast to the tedious multi-page calculations with representations that appeared in the literature of that time. (You can find this proof on page 74 of [11].) During this period, there was also a departmental study group reading Volume 1 of Knuth's *Art of Computer Programming*, and Knuth's treatment of data types, which was informal but suggestive, stimulated a lot of discussion and I think help set the strage for later work on abstract data types.

In fact, I proposed not only a working group, but also the rather grandiose project of providing a uniform, rigorous, abstract treatment of theoretical computer science, similar to what Bourbaki[1] had done for mathematics, but using category theory explicitly, instead of secretly and implicitly, as Bourbaki had done. Surprisingly, this was agreed upon, and Jim Thatcher proposed the name "ADJ" as a (terrible) pun on the title of the book that we planned to write, *A Junction between Computer Science and Category Theory* (substitute "D" for the space between the initial "A" and the following "J" and recall that adjointness is a very important concept in category theory). We also agreed to use ADJ as a name for the set {Goguen, Thatcher, Wagner, Wright}, and to publish all our joint work under that name[2].

While still living in Yorktown Heights, I wrote much of the first third or so of what was supposed to become the first chapter of our book, gently introducing the most basic concepts of category theory, including category, opposite category, and functor, with illustrations from automata and formal languages. This was a lot of fun, but it was also becoming clear that the project was vast, and would

[1]Nicolas bourbaki was the psuedonym of a group of reform-minded French mathematicians who set out to write a series of textbooks on all major areas of mathematics, systematically using an entirely rigorous abstract approach, and emphasizing structure and generality instead of concrete representations for special cases. A good example is the systematic use of various spaces in analysis, from topological to Banach spaces and C^*-algebras. Bourbaki had an enormous influence on the teaching of mathematics, and the group included many of the greatest post-war French mathematicians, and even a few non-French mathematicians. Allegedly, the name was taken from a statue in a town where they held some meetings. I still feel that a similar reform is needed in computer science.

[2]The first instances of the footnote defining ADJ as this set that I have been able to find are in two 1976 papers [12, 13], attached to two different orderings of the four authors. This footnote reads "This *set* of authors is herein, and we hope subsequently will be, referred to by the symbols ADJ." Of course, this was an attempt to defeat the tyranny of the alphabet. A footnote on the third page of [11], attached not to the authors names, but rather to a citation of the report [7], reads "The *set* of authors of this paper is referred to as 'ADJ'; the series referenced here is devoted to an exploration of the 'junction between category theory and computer science."' (By the way, this footnote is *sic*, because it has the subphrases "category theory" and "computer science" in the wrong order.) Thus, perhaps confusingly, the 1977 footnote implies that ADJ was not the author of [11], but was the author of [7]. So "ADJ" was not an acronym, nor even a set, but instead a pun, and also a very curious kind of name, which in 1976 denoted a certain four element set, but in 1977 could only denote that set in certain contexts. I believe that "ADJ" was also been used later to denote other, unequal, sets, but I prefer not to speculate about what this means.

require large amounts of research, as well as enormous amounts of writing. This initial segment of the proposed first chapter appeared as a large IBM technical report [7]. The second third appeared three years later [10], and later still Eric Wagner wrote something like a summary of some of ADJ's work on theories [9]. The trouble was that it seemed desirable, or even necessary, to concentrate on research for a period, writing papers on the many new ideas that we were generating, and coming back to the book sometime later.

One of these ideas was to apply what I had called "initial algebra semantics" and used to formalize Knuth's "attribute semantics" in [3], to many other topics. The key idea was that syntax forms an initial algebra, and its interpretation in any similar algebra of meanings is uniquely determined by the unique homomorphism. Thatcher had some ideas about using an algebra of infinite trees as a more algebraic approach to denotational semantics, and I thought I could prove that such an algebra was initial in a suitable category. This led to a joint paper with Thatcher [5], which eventually evolved into ADJ's fairly well-known paper "Initial algebra semantics and continuous algebras" [11]. This work used initial algebra semantics to explicate McCarthy's notion of abstract syntax, implicitly using the principle that "abstract" means "up to isomorphism" here, just as it does in "abstract" algebra. This idea got carried further, and was also stated more explicitly, in our later work on abstract data types. The paper [11] also applied initial algebra semantics to denotational semantics, tree automata, flow diagrams, and systems of regular equations.

At the end of the summer of 1972, I moved to Los Angeles to take up a post at UCLA. Later that academic year, Jack Dennis gave a lecture at UCLA on some work that Steve Zilles was doing on abstract data types for his Ph.D. thesis at MIT. I got very excited, because I saw that this subject could be developed very elegantly using initial algebra semantics *with equations*. By coincidence, Alan Klinger of UCLA invited me to submit a paper to a conference on computer graphics and data structure, and I took this as an opportunity to do some initial work on abstract data types.

Since this topic was well within the scope of the ADJ project, I decided to put all our names on the paper [8], and I then dutifully sent the draft to IBM for approval, since IBM employees must get all their papers cleared for publication. To understand this story, you need to know that protests about the Vietnam war and the draft were very much on people's minds at this time[3] and you need to know that one of the main motivations for initiality was to replace tedious inductive arguments by diagram chases that exploit the existence of unique homomorphisms (as in the proof of the associativity of substitution).

A key sentence in the introduction of this draft, set in italics on a line by itself, summarized this viewpoint with the slogan:

[3] Non-native speakers of English may also need to be told that "induction" is a somewhat formal synonym for the draft.

Fight induction!

The IBM censors insisted that this had to be removed. Rather than remove the IBM co-authors, I reluctantly agreed to remove the slogan, and thus the paper as published is a bit less well motivated, and a bit less amusing, than it should have been. But this paper did set forth the basic ideas and motivations for the initial algebra approach to abstract data types, as well as to introduce so-called "ADJ diagrams" for signatures.

Raymond Yeh liked this paper, and asked us to do an expanded, expository version for inclusion in a volume on data structuring for a series he was then editing. As there were many points that needed clarification, expansion, and correction in [8], we agreed to try to write such a paper, and worked hard during the summer of 1973 to do so. Unfortunately, the rather tight deadline precluded collaboration with Jesse Wright, because of his blindness. It was quite a strain writing that paper, because we wanted to be very thorough, and to provide ways to prove correctness of implementations, in addition to providing the basic definitions and examples.

Eric Wagner had the wonderful idea of canonical term algebras, and also worked out the complicated but illuminating approach to errors in the appendix. I was trying to figure out how to apply term rewriting techniques, but since that field was still in its infancy and since we had a tight deadline, it didn't get done then, although I did work it out later for the OBJ system (e.g., see [4, 2]). I think this paper [6] finally came out very well, and it may not be unfair to say that it gave the first rigorous theory of abstract data types, and laid foundations for the more general field of algebraic specification that was later to emerge in projects like the specification language Clear [1]. We didn't even try to use the controversial slogan.

Perhaps it was the strain of that paper, plus some others that we tried to write across the Atlantic while I was at the University of Edinburgh on a Senior Visiting Fellowship, that finally broke up the collaboration. In fact, I was able to play only a small part in writing some fascinating but very technical papers [13, 12] on solving systems of equations in ordered algebraic theories, and in the end, quite a lot of what I thought were our most interesting ideas never did get written up thoroughly (such as "$M(T)$"), and some never got published at all, including some lovely results about factorizations in algebraic theories. But ADJ was a wonderful experience for me, and I shall always be grateful to IBM and my ex-colleagues there for the rich stimulation it gave me then, and which has continued to inspire me throughout my career.

References

[1] Rod Burstall and Joseph Goguen. Putting theories together to make specifications. In Raj Reddy, editor, *Proceedings, Fifth International Joint Conference on Arti-*

ficial Intelligence, pages 1045–1058. Department of Computer Science, Carnegie-Mellon University, 1977.

[2] Kokichi Futatsugi, Joseph Goguen, Jean-Pierre Jouannaud, and José Meseguer. Principles of OBJ2. In Brian Reid, editor, *Proceedings, Twelfth ACM Symposium on Principles of Programming Languages*, pages 52–66. Association for Computing Machinery, 1985.

[3] Joseph Goguen. Semantics of computation. In Ernest Manes, editor, *Proceedings, First International Symposium on Category Theory Applied to Computation and Control*, pages 151–163. Springer, 1975. San Fransisco, February 1974. Lecture Notes in Computer Science, Volume 25.

[4] Joseph Goguen. Some design principles and theory for OBJ-0, a language for expressing and executing algebraic specifications of programs. In Edward Blum, Manfred Paul, and Satsoru Takasu, editors, *Proceedings, Conference on Mathematical Studies of Information Processing*, pages 425–473. Springer, 1979. Lecture Notes in Computer Science, Volume 75.

[5] Joseph Goguen and James Thatcher. Initial algebra semantics. In *Proceedings, Fifteenth Symposium on Switching and Automata Theory*, pages 63–77. IEEE, 1974.

[6] Joseph Goguen, James Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In Raymond Yeh, editor, *Current Trends in Programming Methodology, IV*, pages 80–149. Prentice Hall, 1978.

[7] Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. A junction between computer science and category theory, I: Basic concepts and examples (part 1). Technical report, IBM Watson Research Center, Yorktown Heights NY, 1973. Report RC 4526.

[8] Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. Abstract data types as initial algebras and the correctness of data representations. In Alan Klinger, editor, *Computer Graphics, Pattern Recognition and Data Structure*, pages 89–93. IEEE, 1975.

[9] Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. An introduction to categories, algebraic theories and algebras. Technical report, IBM Watson Research Center, Yorktown Heights NY, 1975. Report RC 5369.

[10] Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. A junction between computer science and category theory, I: Basic concepts and examples (part 2). Technical report, IBM Watson Research Center, Yorktown Heights NY, 1976. Report RC 5908.

[11] Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. Initial algebra semantics and continuous algebras. *Journal of the Association for Computing*

Machinery, 24(1):68–95, January 1977. An early version is "Initial Algebra Semantics", by Joseph Goguen and James Thatcher, IBM T.J. Watson Research Center, Report RC 4865, May 1974.

[12] Eric Wagner, Jesse Wright, Joseph Goguen, and James Thatcher. Some fundamentals of order-algebraic semantics. In Andrzej Mazurkiewicz, editor, *Proceedings, Fifth Symposium on Mathematical Foundations of Computer Science*, pages 153–168. Springer, 1976. Lecture Notes in Computer Science, Volume 45.

[13] Jesse Wright, James Thatcher, Eric Wagner, and Joseph Goguen. Rational algebraic theories and fixed-point solutions. In Michael J. Fischer, editor, *Proceedings, Seventeenth Symposium on Foundations of Computing*, pages 147–158. IEEE, 1976.