The Genesis of Attribute Grammars

Donald E. Knuth, Stanford University

I have always been fascinated by the origin of ideas, so I am pleased that the organizers of this conference have asked me to open the proceedings by taking a look back at how the subject began. In other words, I'm glad that my job is to give this conference a "history attribute."

Attribute grammars were born in the exhilarating days of the mid-60s when a great many fundamental principles of computer science were beginning to be understood. An enormous number of ideas were floating in the air, waiting to be captured in just the right combinations that would prove to be powerful sources of future energy.

The human mind is notoriously bad at reconstructing past experiences, so I have tried to make my story as accurate as possible by digging up as many scraps of authentic information from the early days as I could find. People say that nostalgia isn't what it used to be, but I hope that the participants of this conference will be able to share some of the pleasure I have experienced while preparing this retrospective study.

Much of my story takes place in 1967, by which time a great many sophisticated computer programs had been written all over the world. Computing machines had come a long way since their invention in the 30s and 40s; for example, the recently announced Burroughs B6500 was aptly called a "third generation" system [35], and J. P. Eckert was noting a trend toward parallelism in new computer designs [37]. Yet many problems without easy solutions made people well aware that the vast potential of computers was only beginning to be tapped and that a great many important issues were not at all well understood.

One of the puzzling questions under extensive investigation at the time was the problem of programming language semantics: How should we define the meaning of statements in algorithmic languages? Dozens of experts had gathered in Vienna in 1964 for a conference on formal language description, and the prevailing mood at that conference is well summarized by the remarks of T. B. Steel, Jr.: "Improvements in programming language description methods are imperative … I don't fully know myself how to describe the semantics of a language. I daresay nobody does or we wouldn't be here" [31]. Steel edited the proceedings of that meeting, which include lively transcriptions of discussions among the participants [30]. None of the presented papers had much connection with what we now call attribute grammars, although some hints could be found in the explorations of Čulík [4]. At the closing session, Fraser Duncan remarked, "We have not actually got all or perhaps any of the ultimate answers. But we have a lot of ideas of where to look and where not to look" [6].

All of the known methods for defining the meaning of computer programs were based on rather intricate algorithms having roughly the same degree of complexity as compilers, or worse. This was in stark contrast to Chomsky's simple and elegant method of syntax definition via context-free grammars. As Dr. Caracciolo said, "How simple to realize [semantic correctness] if you write a procedure. The problem is, however, to find a metalanguage for doing that in a declarative way, not in an operational way" [3].

There was one bright light on the horizon, a notable exception to the perceived difficulty of semantic definition: Comparatively simple languages such as arithmetic expressions could be understood by a mechanism proposed in 1960 by Ned Irons [9; see also 10, 11]. If we know the meaning of α and the meaning of β, then we know the meanings of such things as $\alpha + \beta$, $\alpha - \beta$, $\alpha \times \beta$, α/β, and (α). Thus the meaning of an arbitrarily large expression can be synthesized in a straightforward way by recursively building up the meanings of its subexpressions.

Simple subexpressions like 'x' still remained problematical, because of their context-dependent meaning. For example the ALGOL 60 program

begin real x; $x := x$ **end**

was implicitly supposed to be parsed by treating the final appearance of x as an $\langle$arithmetic expression$\rangle$, while the analogous x in

begin Boolean x; $x := x$ **end**

was to be parsed as a $\langle$Boolean expression$\rangle$. The official syntax in [28] allowed several distinct parses for both programs, but only one of them was considered to be semantically valid. Furthermore, a program such as

begin real x; $x := y$ **end**

could also be parsed in several ways, from a syntactic standpoint; but none of those parse trees was legal from a semantic standpoint, because the undeclared variable y was neither an $\langle$arithmetic expression$\rangle$ nor a $\langle$Boolean expression$\rangle$. Therefore Irons developed his syntax-directed approach by including mechanisms to reject certain parse trees based on declarations that the parser had found earlier [12].

The prevailing view (see [8]) was that strings of tokens like '**begin real** x; $x := y$ **end**' were not members of the language ALGOL 60, because such strings had no semantically valid parse tree.

In retrospect, we can see why this approach had evolved; syntax was much better understood than semantics, so it acquired additional responsibilities. People found it convenient to incorporate semantical information, about such things as types, into syntactic categories.

My own private opinion was different; I regarded ALGOL 60 as a context-free language defined solely by ordinary productions. To me the "meaning" of a program like

begin real x; $x := y$ **end**

was clearly defined: "Improper assignment of an undeclared variable named y to the real variable named x." But I had no good ideas about how to define such meanings in a simple and systematic way that would generalize nicely.

Besides, I was far too busy with another project; I couldn't think about doing anything new. In the summer of 1962 I had begun to write a book called *The Art of Computer Programming* [14], and I had finished the hand-written draft (some 2500 pages) in June of 1965. By October of that year I had typed the first of its 12 chapters, and at the rate I was going it seemed almost certain that I would be finished typing the entire manuscript by the summer of 1968.

And indeed, I would have finished by then if Computer Science had only cooperated and stood still; and if I hadn't tried to maintain a complete coverage of new ideas as they were developing. I failed to anticipate the fact that the field would grow explosively, that thousands of brilliant people would continually be inventing ever deeper and more important ideas, faster than I could type.

I did realize, of course, that a lot was going on; therefore I agreed to be on the editorial boards of both *Journal of the ACM* and *Communications of the ACM*. In fact, I plunged into those jobs with enthusiasm, because editorial work gave me a chance to learn the new ideas as they were being born. I wrote numerous lengthy critiques of articles that were submitted, trying to improve the quality of the publications as best I could; during the first nine months of 1966 I refereed 21 papers, and I'm sure that the hours I devoted to that work were well spent. My "territory" at the time was called Programming Languages [36]. In March, 1966, I wrote a letter to *Communications of the ACM* editor Gerry Salton making it plain that I was enjoying myself. I said: "The job of being editor in a subject area that is rather new and not too clearly defined is very educational since I am forced to do a lot of reflection on goals and motives which I would never otherwise have done." Thus I had plenty of opportunity to think about thorny issues such as semantics that were much discussed at the time.

Still, I made absolutely no attempt to solve those problems myself. I was plenty busy completing old tasks and keeping up with what other people were doing. When I was on campus at Caltech I had classes to teach and a backlog of editorial work to process; when I came home at the end of the day, there was plenty of typing and revising to do on *The Art of Computer Programming*. At the beginning of 1967 I was beginning to type Chapter 4, which by then was scheduled to appear as the second half of Volume 2 in a projected seven-volume series; the galley proofs of Volume 1 were due to begin arriving in March. Moreover, I was the proud father of a 17-month-old son and a 19-day-old daughter. Research was the farthest thing from my mind.

But I was an ACM Lecturer that year [38], and I spent the middle of February visiting nine campuses and giving sixteen lectures. During that trip I had some fateful encounters; and presto! Attribute Grammars were born.

My first stop was Cornell, and I spent the first weekend staying at Peter Wegner's home in Ithaca, New York. Many details of those two days (February 4–5, 1967) still remain fresh in my mind. For example, I went with Peter to a synagogue on Saturday, and he went with me to a church on Sunday. We hiked outside the city in a beautiful river valley that contained many frozen waterfalls. But mostly we talked Computer Science.

Peter asked me what I thought about formal semantics, and I said I liked Iron's idea of synthesizing an overall meaning from submeanings. I also said that I liked the way other people had combined Irons's approach with a top-down or "recursive-descent" parser; Irons had used a more complex parsing scheme. In particular, I was impressed by the metacompiler of Val Schorre [29] and by the more powerful (though less elegant) "transmogrifier" of Bob McClure [27]. I told Peter about my draft manuscript for Chapter 12 of *The Art of Computer Programming*, which included an example system of a similar kind, called `TROL` ("Translation-Oriented Language"). Programs in `TROL` were, in essence, collections of recursive subroutines corresponding to the definitions of nonterminal symbols in a given language whose compiler was being described. Every such nonterminal could have a variety of attributes of types integer, Boolean, string, or link (i.e., reference); the values of these attributes would be synthesized as parsing proceeded. Context-dependent aspects of semantics were handled in the traditional brute-force way by maintaining global data structures, including symbol tables that allowed dynamic insertion and deletion of ⟨identifier, value⟩ pairs as parsing proceeded. I had prepared two examples of `TROL` programs for the book, one for a procedural language called `3.5-TRAN` and another for `TROL` itself. (The first of these compiled `3.5-TRAN` source programs into `MIX` code; the second compiled `TROL` source programs into `TROLL`, an internal representation of `TROL` code used by a special interpretive routine.)

I probably also told Peter about a significant paper by Lewis and Stearns that I had recently been handling for *Journal of the ACM.* That paper [25] extended the automata-based theory of languages from syntax to semantics, under the assumption that semantics was defined by combination of synthesized string-valued attributes.

Thus, in general, my answer to Peter's question was that the best way I knew to define semantics was to use attributes whose values could be defined on a parse tree from bottom to top. And I said that, unfortunately, we also needed to include some complicated ad hoc methods, in order to get context-dependent information into the tree.

So Peter asked, "Why can't attributes be defined from the top down as well as from the bottom up?"

A shocking idea! Of course I instinctively replied that it was impossible to go both bottom-up and top-down. But after some discussion I realized that his suggestion wasn't so preposterous after all, if circular definitions could somehow be avoided. (The title of my present paper should really be, "The Revelation of Attribute Grammars.")

Well, we didn't pursue the matter, because we had many other things to talk about. But in retrospect, it is clear that a large share of the credit for inventing attribute grammars should be attributed to Peter Wegner, because he was the one who first suggested the concept of inherited attributes. (I've often been kicking myself for failing to acknowledge his important contribution in my first papers on the subject.)

I kept thinking about the combination of inherited and synthesized attributes at odd moments as my trip continued, but I never had a chance to write anything down on paper. A few days later—I think it was on Wednesday, February 15—I found myself at Stanford University, in a conference room with half a dozen or so graduate students of Computer Science. This was a "meet the students" session scheduled after my regular lecture. And one of the students (probably Sue Graham) asked me what I thought was the best way to define the semantics of programming languages. Wow! This was a chance to try out the thoughts I'd been formulating, after Peter had planted that idea in my mind. And there was a blackboard at hand, with plenty of chalk. So I marched up to the board and started to write down a definition of a language very much like what later became Turingol [15]. I have no idea if what I said made any sense to the students, since I must have had to go back and erase a lot of things before getting them right; and I doubt that anybody took notes that day. But I do recall that the first time I ever constructed a nontrivial attribute grammar was during that on-line meeting with students.

Soon I was back home in Southern California, and of course there was no time to write down any of these thoughts. I did, however, take a few minutes to mail a copy of my *curriculum vitæ* to George Forsythe, the chairman of Stanford's department, since I had enjoyed my visit there most of all. (I also wrote a letter to Bob Floyd, who was then at Carnegie-Mellon, saying that I was seriously considering a move to Stanford, or perhaps Cornell; I was hoping that he felt the same way, so we could be together.)

The next chapter in the story of attribute grammars took place in France, of all places. My recollections here are somewhat hazy, because I haven't been able to find any of the related correspondence. But I know that I came to France in May, a week before going to the Simulation Languages conference in Norway [2]. Maurice Nivat had invited me to speak at the Institute Blaise Pascal, and someone (probably Louis Bolliet) had arranged for me to spend two days in Grenoble. On one of those two days—I think it was Friday, May 19—I was once again asked my opinion about semantics, and I made an informal presentation of a definition for Turingol to about a dozen people. That was the second time I had had occasion to write down an attribute grammar.

On the following day I went to Zandvoort, Holland, where a meeting of the IFIP ALGOL working group was being held. About 50 people were sitting around a large round table, making decisions about ALGOL X (which had previously been called ALGOL 66 and which eventually became ALGOL 68); as you know, this language was an extensive development of ALGOL 60, based on Aad van Wijngaarden's method of semantic definition via recursive interpretation of texts [33]. My main impression of that meeting was that Chris Strachey's prediction from 1964 was unfortunately being fulfilled: He had said, "In anything like a complicated language . . . , the process of defining a compiler for it is extremely elaborate. The details of the special compiler written by the committee are bound to be confined to the people who wrote it, and they will be the only people who understand it" [32].

Back home in California, I once again was completely preoccupied by editorial work and by fatherhood; there was no question of looking any further into semantics and such. I was getting into some very difficult parts of Volume 2 that I decided were necessary additions to my first draft. (For example, I had gone to a combinatorics conference in April [1], where I had learned about Berlekamp's amazing new method for factoring polynomials modulo 2.) Still, I returned a batch of galley proofs to Addison–Wesley on June 5 saying, "I had a very nice rest on this trip to Europe and I am ready to dive into work this summer."

But those months of June and July, 1967, were the worst of my entire life. While I was working out the answer to a new exercise for my book (exercise 4.5.2–18), I suddenly had to make an unscheduled trip to the hospital. The ensuing dark days are best summarized by quoting from a letter I sent to my publishers at the end of July:

> . . . my "iron stomach" sprung a leak. I was forced to go five weeks without doing any work (all I could do was eat Jello, plain, and read Agatha Christie stories). After this I began to feel fine again, probably since I have always liked Agatha Christie. So then like a fool I began to work once more at a strenuous pace, and in two more days I was down again. It is clear that I have been trying to do more concentrated work than my body will stand, so I will have to systematically decrease my work load. In the immediate future (say the next month) I should relax except for about five hours a day when I can concentrate on my writing, etc.; after this I should be knit together enough to work say nine or ten hours a day, but never go back to the 14-hour schedule I had been keeping.

On September 11, 1967, I reluctantly resigned from my position on the *ACM Journal* editorial board.

Meanwhile, other responsibilities were still in place. I had agreed to give a week's lectures at a NATO Summer School near Copenhagen in August, and my plan was to come a week early so that I would have time to figure out what to say. Well, I arrived in

Denmark on August 6 as planned, only to learn that Klaus Wirth had become seriously ill and was forced to cancel his lectures; therefore I was asked to take his place. I agreed to begin on Wednesday afternoon, although that gave me only 2.5 days to prepare. My announced title was "Top-Down Syntax Analysis," and I was hoping that I would be able to prove some useful new theorems in time to present them in the lectures. In order to keep my stomach calm, I decided to work outdoors, sitting under some stately pine trees in the woods near the campus at Lyngby. Thank goodness, my luck held; the concepts of LL(k) took shape as I had hoped they would, during those 2.5 days in the Danish woods. I lectured all day Thursday and Friday without serious side effects, stomachwise; but a bug in the theory surfaced on Friday afternoon, just as I was finishing. I fixed everything on Saturday morning, and finally began to enjoy the sights of Denmark. (Notes from those lectures were eventually published in [19].)

You might think that I now had a week free to think about semantics, but that was not the case. I was on my way to another conference at Oxford, where I was scheduled to speak about another topic entirely. I had stumbled across this other idea (which has become known as the "Knuth-Bendix algorithm") while teaching an undergraduate class in 1966. Peter Bendix (a student in the class) had implemented the method and we had carried out numerous experiments; so the research was all done and I merely had to write up the paper [16]. That was how I spent my "free" week in Denmark.

I remember talking to John McCarthy about sensory overload during my visit to Stanford in February. I said I was in a dilemma because I was sitting on two ideas I thought were going to be important (namely, the Knuth-Bendix algorithm and the definition of attribute grammars), but I couldn't investigate either one carefully because I had to devote all my spare time to *The Art of Computer Programming*. I asked his advice: Should I publish them in embryonic form (and let other people have the fun of developing them), or should I hold back awhile until I had time to study them thoroughly, then present more mature concepts? John said I should wait until I had time to work everything out personally, otherwise there was a danger that the ideas would be misunderstood and distorted. I guess I didn't follow his advice very well, but I did at least take time to explain the ideas as well as I could when I did publish them. And I've been lucky that both ideas seem to have inspired many other people to do things I could never have accomplished.

Although attribute grammars remained at the back of my mind for several months, my next chance to think seriously about them didn't come until I was away from home again—this time at a SIAM conference in Santa Barbara, California, at the end of November. Although the conference record lists me as one of the participants [39], the truth is that I spent almost the whole time sitting on the beach outside the conference hotel writing a paper about "semantics of context-free languages" [15]. That paper explained everything I knew (or thought I knew) about attribute grammars. I spent the first day working on a test for circularity; after rejecting three obviously false starts, I thought I had found a correct algorithm, and I didn't try too hard to find fault with it.

So that's the story of my paper [15], which was effectively written in Ithaca, Stanford, Grenoble, and Santa Barbara. (The published paper [15] says 'Received 15 November 1967'; that must be a misprint for December.) I've presented this background information in some detail because it suggests that research institutes may not be the best places to do research. Perhaps new ideas emerge most often from hectic, disorganized activity, when a great many sources of stimulation are present at once—when numerous deadlines need to be met, and when other miscellaneous activities like child-rearing are also mixed into the agenda. (On the other hand I'm quite content to be leading a much more orderly life nowadays. One attack of ulcers is enough for me.)

I moved to Stanford and began teaching there in the fall of 1969. Now I was officially a computer scientist instead of a mathematician. Several Stanford students asked me to give them further examples of attribute grammars, and in February 1970 I received a letter from Erwin Engeler requesting the same thing. In April and May, I taught a graduate course about compilers based on my notes for Volume 7 of *The Art of Computer Programming*; it was the one and only time in my life that I have ever taught such a course. My class notes give no indication that I lectured at all about attribute grammars; there was very little time, less than nine weeks altogether, because classes were often disrupted by political crises in those exciting days. I did, however, devote a number of lectures to the `TROL` language, which was the basis of the main homework assignments. (Two of the students submitted an interesting term paper containing a critique of `TROL` and a suggested successor called `STROL`. Their names? Ron Rivest and Bob Tarjan.)

My work pattern in 1970 was somewhat different from the frenetic pace of 1967. At home, I continued to type away at *The Art of Computer Programming*; I was then finishing up the treatment of Shellsort, in Section 5.2.1 of Volume 3. But I would go to sleep when I got tired. At school, I was now expected to do some original work, in addition to classroom teaching, in order to please my research sponsors. Thus, I was no longer confining my research activities to odd moments when I was away from home (although it is true that paper [17] was written on another California beach).

It was hard for me to do anything creative in my office, with the phone ringing and people dropping in all the time; so I found a few quiet hideaways on the Stanford campus. In particular, I spent three or four pleasant days sitting under an oak tree near Lake Lagunita, writing the requested chapter for Engeler's book [18]. Sitting under this tree in springtime was an ideal way to get into the right mood to write about an intermediate language TL/I for Turing machines, and to come up with the philosophical discussions in that paper. (I had previously corresponded with Clem McGowan of Cornell about the semantics of lambda expressions, hence that part of [18] could be gleaned from older notes.) Incidentally, Stanfordians will appreciate the fact that baby caterpillars kept falling on me as I was sitting under the oak tree.

It is clear from reading [18] that I was still unaware of the serious error in the circularity test of [15] when I wrote the new paper. Indeed, I returned the galley proofs of [18] to the printer on July 28; then on August 6, I received a letter from Stein Krogdahl in Norway, containing an elegantly presented counterexample to my circularity algorithm. (His letter had come by surface mail, taking six weeks to reach me, otherwise I could have alluded to the problem in [18].) I soon found a way to patch up the difficulty, but the worst-case running time of the new algorithm was now exponential instead of polynomial. I immediately sent an errata notice [20] to the publishers of my original article, including also a correction to equation (2.4); Jiří Kopřiva had written to me in 1968, pointing out an oversight in that formula.

Alas, it is impossible to get everything right, even when (or especially when?) we are writing about how to make careful definitions.

Sometime during 1970—I cannot recall whether it was in the winter, spring, or fall—John McCarthy challenged me to a public debate about how semantics ought to be defined.* (John did this for fun; he loves to debate things. See, for instance, [26].) Unfortunately, I saved no notes of what happened during our friendly hour-long confrontation; but I do recall that our main point of disagreement concerned inherited attributes, which John thought were unnecessary. His recommended alternative was, in essence, to associate parameters with the nonterminal symbols of a grammar; these parameters could bring contextual information down to the leaves of the tree. For example, instead of a nonterminal symbol E for expressions, we could have $E(s)$, where s was an appropriate symbol table for declared identifiers allowed within the expressions. I had never given much thought to such an approach, so I had no counterexamples handy to show why inherited attributes would be more powerful and/or more natural than a parameter mechanism. I don't think either of us "won" the debate; but I do remember promising myself afterward never to engage in such a thing again, because I have never enjoyed verbal argumentation.

My own work on attribute grammars had to become dormant soon afterward, because there were too many other things to do. (My chief activity during the summer of 1970, besides the ongoing work on Volume 3 of *The Art of Computer Programming*, was an empirical study of FORTRAN programs [21] carried out with the help of a dozen students and other volunteers.) I gave a talk that November to the participants in a Research Workshop in Grammar and Semantics of Natural Languages that Pat Suppes had organized at Stanford. I'm not sure exactly what I said, nor have I any evidence that my talk made a great impression on anybody in the audience, but my views at the time can be faithfully summarized by the following abstract I sent to the participants in September:

---

* While preparing this paper, I tried to discover the date of our debate by looking at John's computer files from that era. Thanks to the modern-day wizardry of Martin Frost and Joe Weening, it is still possible to reconstruct many of his electronic files from 1969 and 1970; but all we could find relevant to attribute grammars was an entry within a file called `TOREAD`, last updated 26 October 1970, in which my paper [18] was listed sixth from the top.

> My motivation for this work was entirely directed to semantics of computer programming languages, but there is reason to believe that the approach is useful for natural languages as well.
>
> The enclosed pages introduce the idea, which is essentially very simple. For natural languages the "synthesized attributes" would be things like number, gender, denotation, etc., while the "inherited attributes" would be things like the meaning of prepositions based on context. This approach to semantics also seems to make it possible to simplify the syntax of a language in a fairly natural way.

During my first years at Stanford I supervised two students whose Ph.D. theses explored the large-scale application of attribute grammars [7, 34]. But I soon discovered that I could no longer participate adequately in widely different parts of Computer Science all at once, so I began to concentrate on mathematical analysis of algorithms [22].

Of course it's been a treat for me to see how attribute grammars have grown in popularity during the past 20 years. When I first learned about the astonishing results of Jazayeri, Ogden, and Rounds [13]—that circularity can always be tested in A^n steps and that every correct algorithm needs to perform at least $B^{n/\log n}$ steps on infinitely many examples—it blew my mind. This made circularity testing one of the first "natural" problems to have provably exponential complexity. Never would I have dared to conjecture such a remarkable theorem. Then I learned that even my incorrect circularity algorithm was turning out to be useful, because it provides a test for "strongly non-circular" grammars (see [5, page 18]).

In 1977 I began to work on a language for computer typesetting called TEX, and you might ask why I didn't use an attribute grammar to define the semantics of TEX. Good question. The truth is, I haven't been able to figure out *any* good way to define TEX precisely, except by exhibiting its lengthy implementation in Pascal [23]. I think that the program for TEX is as readable as any program of its size, yet a computer program is surely an unsatisfying way to define semantics. Still, it's the best I have been able to do. Moreover, I don't know any way to define any other language for general-purpose typesetting that would have an easily defined semantics, without making it a lot less useful than TEX. The same remarks apply also to METAFONT. "Macros" are the main difficulty: As far as I know, macros are indispensable but inherently tricky to define. And when we also add a capability to change the interpretation of input characters, dynamically, we get into a realm for which clean formalisms seem impossible.

During the ten years I was working intensively on TEX, I occasionally ran into people who said that they were interested in attribute grammars. But when I saw the book *Attribute Grammars* by Deransart, Jourdan, and Lorho [5], I was astonished. I could hardly believe the fact that its bibliography cited about 600 relevant papers. Wow; surely I would never have predicted such dramatic growth.

I'm delighted, above all, to see that people not only use attribute grammars, and prove deep results about them, they have fun doing so. I looked at about dozen of the papers cited in [5], and in each case I noticed the authors' evident enthusiasm for the work they were doing. Nothing could make me happier.

My own contribution obviously amounts to only a minuscule portion of the many things that have been discovered. Attribute grammars would not have become so widespread so quickly if the concept had not been quite a natural one. Therefore somebody would surely have invented attribute grammars even if Ned Irons, Peter Wegner, and I had never existed. Yet as a writer I'm extremely pleased to have written a few papers that people have found inspiring.

Let me close by taking this opportunity to correct a few residual errors in those papers. First, in [15]: On page 137, line 10, 'from (X_{p0}, α')' should be 'from (X_{p0}, α) to (X_{p0}, α')'. There should be more space after paragraph (3) on page 139. There should be another sentence added to the fourth-last line of page 142: 'The idea of inherited attributes was suggested to the author by Peter Wegner.' And the name '*birgitta*' should be '*birgitte*' in two of the examples on page 141. (Those identifiers were named after Peter Naur's daughter Birgitte Naur, whom I met during that summer of 1967.)

Next, in [20]: On page 95, in the second correction for page 141, the words 'newsymbol', 'include', and 'in' should be **boldface**. In the displayed equation for $S(L)$, the labels '1' should be 'l'. On both pages 95 and 96, the notation 'X_{po}' should be 'X_{p0}' (five places). In the statement of the theorem on page 96, '*roles*' should be '*rules*' and 'X_{n_p}' should be 'X_{pn_p}'.

And in [18]: Change 'quadruples' to 'quintuples' on line 2 of page 213. Insert quote marks around 'address' in the description of label(σ) on page 221. Change 'situtation' to 'situation' on line 2 of page 225. Change the second semantic rule of 4.1 on page 232 to 'fin(S):=fin(L)'. And append the following to page 235: '*Acknowledgment.* I wish to thank Clement L. McGowan III for several stimulating discussions, especially for pointing out errors in my original attempts to define the semantics of lambda expressions.'

Finally, change 'which' to 'that' in about 100 places, throughout the articles (see [24, §38]).

Bibliography

[1] R. C. Bose and T. A. Dowling (editors), *Combinatorial Mathematics and its Applications*, Proceedings of a conference held at Chapel Hill, North Carolina, April 10–14, 1967; *University of North Carolina Monograph Series in Probability and Statistics* **4** (1969).

[2] J. N. Buxton (editor), *Simulation Programming Languages*, Proceedings of an IFIP Working Conference held at Oslo, Norway, May 22–26, 1967 (Amsterdam: North-Holland, 1968).

[3] A. Caracciolo di Forino, "On the concept of formal linguistic systems," in [30], 37–51.

[4] Karel Čulík, "Well-translatable grammars and Algol-like languages," in [30], 76–85.

[5] Pierre Deransart, Martin Jourdan, and Bernard Lorho, "Attribute Grammars," *Lecture Notes in Computer Science* **323** (1988), ix+232 pp.

[6] F. G. Duncan, "Our ultimate metalanguage: An after dinner talk," in [30], 295–299.

[7] Isu Fang, FOLDS, a declarative formal language definition system. Ph.D. thesis, Stanford University, 1972; also report STAN-CS-72-329.

[8] Robert W. Floyd, "On the nonexistence of a phrase structure grammar for ALGOL 60," *Communications of the ACM* **5** (1962), 483–484.

[9] Edgar T. Irons, "A syntax-directed compiler for ALGOL 60," *Communications of the ACM* **4** (1961), 51–55.

[10] Edgar T. Irons, "The structure and use of the syntax-directed compiler," *Annual Review in Automatic Programming* **3** (1962), 207–227.

[11] Edgar T. Irons, "Towards more versatile mechanical translators," *Proceedings of Symposia in Applied Mathematics* **15** (American Mathematical Society, 1963), 41–50.

[12] Edgar T. Irons, "'Structural connections' in formal languages," *Communications of the ACM* **7** (1964), 67–72.

[13] Mehdi Jazayeri, William F. Ogden, and William C. Rounds, "The intrinsically exponential complexity of the circularity problem for attribute grammars," *Communications of the ACM* **18** (1975), 691–706.

[14] Donald E. Knuth, *The Art of Computer Programming* (Reading, Mass.: Addison–Wesley). Volume 1, 1968; Volume 2, 1969; Volume 3, 1973; Volume 4, in progress.

[15] Donald E. Knuth, "Semantics of context-free languages," *Mathematical Systems Theory* **2** (1968), 127–145.

[16] Donald E. Knuth and Peter B. Bendix, "Simple word problems in universal algebras," in John Leech (editor), *Computational Problems in Abstract Algebra*, Proceedings of a conference held at Oxford, England, August 29–September 2, 1967 (Oxford: Pergamon Press, 1970), 263–297.

[17] Donald E. Knuth, "The analysis of algorithms," Actes du Congrès International des Mathématiciens 1970, **3** (Paris: Gauthier-Villars, 1971), 269–274.

[18] Donald E. Knuth, "Examples of formal semantics," in E. Engeler (editor), Symposium on Semantics of Algorithmic Languages, *Lecture Notes in Mathematics* **188** (1971), 212–235.

[19] Donald E. Knuth, "Top-down syntax analysis," *Acta Informatica* **1** (1971), 79–110.

[20] Donald E. Knuth, "Semantics of context-free languages: Correction," *Mathematical Systems Theory* **5** (1971), 95–96.

[21] Donald E. Knuth, "An empirical study of FORTRAN programs," *Software—Practice & Experience* **1** (1971), 105–133.

[22] Donald E. Knuth, "Mathematical analysis of algorithms," *Proceedings of IFIP Congress 1971*, **1** (Amsterdam: North-Holland, 1972), 19–27.

[23] Donald E. Knuth, *TEX: The Program.* (Reading, Mass.: Addison–Wesley, 1986), xv+594 pp.

[24] Donald E. Knuth, Tracy Larrabee, and Paul M. Roberts, *Mathematical Writing*, MAA Notes **14** (Mathematical Association of America, 1989), v+115 pp.

[25] P. M. Lewis II and R. E. Stearns, "Syntax-directed transduction," *Journal of the ACM* **15** (1968), 465–488.

[26] John McCarthy, in "General discussion," *Communications of the ACM* **7** (1964), 134–136.

[27] Robert M. McClure, "TMG–A syntax-directed compiler," *Proceedings of the ACM National Conference* **20** (1965), 262–274.

[28] Peter Naur (editor) et al., "Report on the algorithmic language ALGOL 60," *Communications of the ACM* **3** (1960), 299–314.

[29] D. V. Schorre, "META II–A syntax-directed compiler writing language," *Proceedings of the ACM National Conference* **19** (1964), D1.3.1–D.3.11.

[30] T. B. Steel, Jr. (editor), *Formal Language Description Languages for Computer Programming*, Proceedings of an IFIP Working Conference held at Vienna, Austria, September 15–18, 1964 (Amsterdam: North-Holland, 1966).

[31] T. B. Steel, Jr., “A formalization of semantics for programming language description,” in [30], 25–36.

[32] Christopher Strachey, comment on paper by J. V. Garwick, ‘The definition of programming languages by their compilers,” in [30], 139–147.

[33] A. van Wijngaarden, “Recursive definition of syntax and semantics,” in [30], 13–24.

[34] Wayne Theodore Wilner, A declarative semantic definition. Ph.D. thesis, Stanford University, 1971; also report STAN-CS-71-233.

[35] “Burroughs announces B6500,” *Communications of the ACM* **9** (1966), 541.

[36] “Salton announces new departments, editors for *Communications*,” and “*Journal* changes are outlined by Gotlieb,” *Communications of the ACM* **9** (1966), 825.

[37] “On FJCC 66 at the Golden Gate,” *Communications of the ACM* **9** (1966), 885.

[38] “ACM National Lecturers 1966–7,” *Communications of the ACM* **10** (1967), 68–69.

[39] “SIAM 1967 Fall Meeting,” *SIAM Review* **10** (1968), 260–280.

The preparation of this paper was supported in part by NSF grant CCR–8610181.