Symbolic Mathematical Computation 1965–1975: The emergence of a discipline

Robert M. Corless[a,e,*], Arthur Norman[b], Tomás Recio[c], William J. Turkel[d], Stephen M. Watt[e]

[a]*Department of Computer Science, Western University, London, Canada*
[b]*Trinity College, Cambridge, UK*
[c]*Departamento de Matemáticas y Física, Universidad Antonio de Nebrija, Madrid, Spain*
[d]*Department of History, Western University, London, Canada*
[e]*Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada*

Abstract

Today, symbolic mathematical computation is taken for granted as part of the scientific infrastructure, but it has not always been so. This paper provides a historical survey of the discipline's formative decade, 1965–1975, viewed from a 50 year perspective. This span of years saw the evolution from a few specialized programs with naive algorithms to integrated systems with substantial capabilities. We highlight some of the important early figures in the field and the innovations upon which the current generation of systems and algorithms are built. By revisiting a period unfamiliar to most current readers, this survey aims to shed light on once-pressing issues that are now largely resolved and to highlight how some of today's challenges were recognized earlier than expected.

Keywords: Symbolic Computation, Computer Algebra, History of Computer Algebra, Mathematical Software, Algebraic Algorithms

1. Introduction

The modern symbolic computation world is quite large in absolute terms, comprising on the order of a thousand active researchers and a million dedicated users, and even more people who use it occasionally[1]. Symbolic computation has a very large impact through a variety of software packages and problem-solving environments.

The history of any important field is of intrinsic interest, and every researcher in it could benefit from knowing the main currents of its history. Such knowledge also has instrumental value in that knowing the history can prevent new projects from repeating earlier mistakes and can help overcome similar obstacles. As the ironic saying goes, 'Six months in the lab can save you three days in the library.'

Finally, the history of symbolic computation is a crucial component of the history of mathematical thought more broadly since it was one way that ideas of abstraction and axiomatic thinking reached much larger audiences from the mid-20th century onward [57, 154].

This paper tells part of the story of the history of symbolic computation in a way that addresses these desires. We have spent some time surveying, reading, and digesting the early literature on symbolic computation, and here we present some highlights of the period 1965–1975. However, this paper is not comprehensive. We welcome discussion, especially of things we have missed. If you have knowledge that has passed us by please get in touch so that as full a story can be told as possible.

*Corresponding author
Email addresses: rcorless@uwo.ca (Robert M. Corless), acn1@cam.ac.uk (Arthur Norman), trecio@nebrija.es (Tomás Recio), william.j.turkel@gmail.com (William J. Turkel), smwatt@uwaterloo.ca (Stephen M. Watt)

[1]These numbers are not rigorous and may not be correct even within a factor of ten, because they depend on the exact definition of what it means to be an "active researcher" in the field, or a "dedicated user." Still, by roughly counting the distinct attendees at various conferences, the number of authors of JSC articles, and the number of developers of various systems (not forgetting the large group of Symbolic Toolbox developers at The Mathworks), and acknowledging the 2018 public claim of "100 million users of GeoGebra," which we asked the GeoGebra team about and they told us that as of 2025 the number of downloads was over 144 million, we think our estimates are plausible.

When SIGSAM was being founded those involved had to decide what to call the new group. Sammet in [145] says "The phrase *formal algebraic manipulation* applies to the computer processing of formal mathematical expressions without any particular concern for their numeric values." Other writers wondered if formal logic should be included. After discussion they ended up with "Symbolic and Algebraic Manipulation," well aware that some people would have used other mixes of words. Similarly ISSAC ended up making "Symbolic and Algebraic Computation" definitive at least for the first few decades. No group wanted the choice of names they had made to be used to exclude interesting work with relevance to their core interests, and gradually the body of publications by each group defined by example what the field was seen to encompass at the time.

By 1972, the field (whatever its name was) was beginning to be incorporated into the curriculum: see [36], which was cross-published in the SIGCSE (Computer Science Education) Bulletin as well [35]. In there we find the field defined as follows:

> *Symbolic Mathematics* is the study of algorithmic procedures for manipulating symbols that represent mathematical objects.

We think most readers today would agree that this definition is not satisfactory, because (for instance) on its face it includes numerical mathematics.

Things are a bit more clear-cut nowadays, and the name of the Journal of Symbolic Computation is meant to be inclusive, as discussed in detail in [29]. The *Computer Algebra Handbook* [78] also tries to define what topics are relevant to computer algebra, in the more than one hundred and fifty pages and twenty sections of its Chapter 2, but even there some topics that *could* have been included—such as interval arithmetic and automatic differentiation—were not included, while some of the topics that *were* included might be controversial: MathML, for instance, is extremely useful for sharing mathematics, but is it "Symbolic Computation"? It is still possible to disagree on whether a topic should be included or not.

However, we do not attempt here to define what is, and what is not, a topic relevant to Computer Algebra even for the time period we consider, and instead we simply admit that we omit some topics in this paper. We used "Symbolic Mathematical Computation" in the title, to make it a bit clearer for outsiders. At one end of the spectrum have been groups that have built software to be applied in a range of mathematical and engineering domains. At the other end have been those concerned with theory. To some extent, this can be seen as the continuation of earlier work to rebuild mathematics from a constructivist standpoint. The mix of tensions and cross-pollination across the different approaches is part of what gives the subject and its associated conferences some of its flavour.

We have organized our report chronologically in sections, more-or-less ending in 1975 (about a half century ago). Activity in each period had its own distinctive dynamic. Within each section we use the device of classifying contributions as those for software systems, applications, algebra and formal proof.

By and large we *exclude* numerical computation here, except for multiple precision and except insofar as it was used for exact computation. See the impressive work [17] for an 800-page journey through just the history of numerical linear algebra, as a partial recompense for this exclusion.

Our focus is on symbolic mathematical computation for programmable computers. We highlight key algorithmic developments in this context.

We have chosen to frame our work by focusing first on the contributions of Jean E. Sammet (1928–2017), who contributed signally to the early symbolic computation world [52] and took prime responsibility for the SYMSAC'66 conference that was celebrated in Guanajuato as having grown to become ISSAC. Among her further achievements, she was the founder and first Chair of SIGSAM, the ACM Special Interest Group on Symbolic and Algebraic Manipulation. She was the lead designer of Formac, which was the first commercially successful programming language for computer algebra, and was instrumental in the development of Cobol. She later became the first female president[2] of the ACM itself. See figure 1.

Sammet was also a historian of computing. "From childhood on, I hated to throw papers away. As I became an adult, this characteristic merged with my interest in computing history. As a result, I created important files and

[2]This is not so much indicative of Sammet's ability or force of personality, but instead an indictment of the times. Some of the language of those times will necessarily creep into this paper: for instance, Sammet became the first "Chairman" of SICSAM, later SIGSAM. This may read oddly to more modern sensibilities.

Figure 1: Jean E. Sammet, the founding Chair of SIGSAM and General Chair and Program Chair of SYMSAC '66

documents of my own, and became concerned with having other people publish material on their important work so the facts (rather than the myths) would be known publicly." [67] She wrote an encyclopedic book [145] on the early history of programming languages (concentrating on those invented in the USA), including those for computer algebra, and later wrote a condensed paper on the same subject, together with a look to the future [146]. As well as her having started the SYMSAC conference series and founding key early groupings and publications, her output has helped us to see how those active in the field saw it at the time, and in particular what they viewed as part of the subject: so what we report here is less tainted by our modern views than might otherwise have been the case.

For more details of her life, consult the IEEE Oral History interview[3] of her by Janet Abbate [1]. We learn that she never completed a PhD, leaving graduate school with only a Masters' in abstract algebra (from a pure mathematics department where computers were held in contempt—which is the word she used). She tried to become a math teacher, but met with insurmountable trivialities, and instead went into industry, first at Metropolitan, then Sperry Gyroscope, then Sylvania Electric, and finally IBM. It was in industry where she learned to make wiring diagrams for punch-cards, which she found she really enjoyed, and to use analogue computers for military engineering problems, which she didn't much care for. Eventually digital computers came into play. The interview gives her explanation of how she taught herself to program, working at Sperry Gyroscope, and indeed taught some of the very first college courses in programming, before she took a position at IBM:

> My boss's boss, my manager, came over to me one day and said, "By the way, do you know we're building a digital computer?" I said, "I've heard of it. I'm not sure I know what that means." He said, "We are building a digital computer, because we can see that that is the wave of the future." Because this was a company that made its living on contracts with the Federal Government, and in particular with the military, so they could see, in 1955, that that was going to be the wave of the future. He said, "We're building this thing. Do you want to be our programmer?" I asked the obvious question: "What is a programmer?" And he said, "I don't know, but I know we need one!" [laughs.] I looked at him, and I said, "Well, is this anything like working with punch-cards?" And he said, "I'm not sure, but I think it might be." So I thought, "Well, that punch-card stuff was fascinating. I'm not too inspired with this submarine stuff. I should at least give this a try." So I became the programmer. No books, no manuals,

[3] Jean Sammet, an oral history conducted August 2001 by Janet Abbate, IEEE https://ethw.org/Oral-History:Jean_Sammet. The ETHW has a collection of more than 900 oral histories in electrical and computer technology which can be accessed via http://ethw.org/Oral-History:List_of_all_Oral_Histories.

> no instructions, no nothing—and engineers who somehow thought that machine was going to run itself and didn't want a programmer. Fortunately, my boss, who was a very nice and very smart engineer, knew they needed one, even if he didn't quite understand what that meant.

The rest of the interview is well worth reading.

2. 1964 and before

Of course there is a significant prehistory of computer algebra. In [159] we find a nuanced discussion of the role of Charles Babbage himself as a potential ancestor, expressing himself in writing on the topic as early as 1836. Ada Lovelace's later writings (1846) amplify Babbage's musings, and make it clear that the analytical engine was being viewed with a potential for algebraic, not just numeric, computation. See [14] and [104] for more discussion of this prehistory.

By the 1930s mathematicians had developed a concept of an "effective procedure" for completing a task, even though computers as we know them were not available. The 1926 paper of Grete Hermann [90], translated to English in 1998 [91], was cited in [28] (the table of contents of which is in [27]) as being of special importance. These century-old works really set up the idea of what could be done algorithmically, and the biggest result associated with it was Gödel's Incompleteness Theorem which effectively dashed the hopes of those seeking to rebuild all of mathematics on constructive foundations. A key point to make here is that in deriving that result, Gödel publicised the fact that any formula could be encoded using a number.

The Turing machine was defined in the same decade. It established a framework for mechanical computation that could handle mathematical equations and a desire to solve them exactly or prove properties of them. It also suggested using one computational model to simulate another: in effect introducing the concept of an "interpreter". In the same time-frame, the American logician Alonzo Church introduced the lambda calculus as a model of computation. That the lambda calculus and Turing machines provide equivalent computational power is the Church-Turing thesis. Church also proved that problem of deciding validity of formulas in first-order logic is unsolvable [39, 38], which has implications throughout computer algebra, including subsequent work on the decision problem [53] and later Richardson's theorem on the undecidability of symbolic real expressions [136]. Of course, all of that work was theoretical and much of it did not see general algebraic transformations and simplifications as the main objective, but it provided a basis for those who came later.

By the late 1940s and through the first half of the 1950s an increasing number of mathematicians became able to access electronic computers. Several of them had prior experience using electromechanical calculators to compile tables of special functions or to predict the trajectories of shells: work that was numerical in style. But some looked at problems that might be described as pure rather than applied mathematics. Perhaps the main focus was on support for logic, inference, and proof checking. Today these are still official topics of ISSAC, but in the early years schemes set up to handle them became directly applicable to symbolic elementary algebra.

A notable contribution was IPL (Information Processing Language) by Allan Newell, John Clifford Shaw, and Herbert A. Simon (c. 1956) [4], which introduced list-based symbolic data structures, recursion, and dynamic memory allocation [145, 126]. IPL was developed in close connection with the Logic Theorist, one of the earliest artificial intelligence programs, which demonstrated that machines could discover proofs of theorems in symbolic logic [127]. In that system, logical expressions were represented as structured symbolic objects with an explicit tree organization (as described in the original report), and proof search was guided by heuristic strategies rather than exhaustive enumeration. These requirements motivated the use of flexible symbolic representations together with operations for dynamically constructing and traversing such structures. The Logic Theorist operated on formulas drawn from *Principia Mathematica* and was able to prove a substantial subset of its early theorems [127]. Thus IPL can be seen not merely as a programming language innovation, but as part of an integrated approach to symbolic reasoning, in which representation and control mechanisms were designed to support heuristic search in spaces of mathematical expressions. Newell and Simon demonstrated their automated reasoning program at the 1956 Dartmouth Summer Research Project on Artificial Intelligence, which was co-organized by John McCarthy.

[4] See the description of the contributions of Newell, Shaw, and Simon at the History of Information website. https://www.historyofinformation.com/detail.php?id=742

FORTRAN made its first appearance in an IBM memo dated November 10, 1954, and FORTRAN II was released in June 1958. See [145] for a detailed discussion. As will be seen in the next section, FORTRAN later became an important base for mechanized symbolic computation. But from its inception (indeed from the time of AUTOCODE some years earlier) the issue of parsing algebraic formulae had been addressed. From technical reports published even some years later it is clear that this was a challenging task for quite some time. It is hard to have an algebra system without a way for the system to accept reasonably natural mathematical input! Indeed some of the very earliest attempts used what now seem like bizarre tabular notations for both problem presentation and delivery of results.

ALGOL was developed in the period 1955–1957 by work started by a committee struck by GAMM[5] [145, p. 173]. COBOL was developed extremely rapidly, taking just the last six months of 1959 [145, p. 331]. LISP, very much a successor to IPL, originated at around the same time (the key publication [114] was in 1960), and those working close to McCarthy started to use it for a range of symbolic calculations.

As with IPL, some of the very earliest activity concentrated on logic rather than algebra. Efficient list processing, the first LISP compiler (1962), extended precision integers and floating point values (including elementary function evaluation), and garbage collection would eventually build on this to provide necessary infrastructure for symbolic algebra.

Somewhat later, the first in the SNOBOL family of string processing programming languages was initiated [65, 66] at Bell Telephone Laboratories. This emerged from the work on SCL [105, 81], "a Language for Symbolic Communication," used for symbolic integration, factoring of multivariate polynomials and the analysis of Markov Chains [81]. Symbolic computation has continued to be one of the intended application areas of the SNOBOL languages. For a summary of the early history of programming languages, see [146], which has a beautiful map showing lines of descent.

The state of the art of symbolic computation software in the period covered by this section has been captured in Sammet's survey article [143]. On the algorithmic side, of particular note is the emergent interest in faster methods for multiplication by Karatsuba [96] and leading to Toom-Cook multiplication [44, 157].

The situation at the end of the 1950s can be looked at in two quite contrasting ways. Almost nothing that we would now view unambiguously as an ISSAC topic had yet been published. But several projects had begun and a very substantial body of fundamental work would lead to tremendous growth the following decades. From that perspective the bounty of the 1950s included:

1. Computers of increasing power and reliability became fairly broadly accessible to people wishing to develop advanced applications;
2. High level languages rather than machine code became a practical strategy for developers, with some hope for cross-platform portability. FORTRAN and LISP were the ones that were widely used then and have lasted, but ALGOL existed and has influenced almost everything since, and at the time string processing such as SNOBOL were seen as core technology for symbol manipulation;
3. List and tree data structures with automatic storage management was an understood concept – either built into the language used or implemented as part of the program being written;
4. There were links between practical and theoretical work and workers relating to algorithms and computation. This spanned cost analysis of algorithms where today the results would be reported in big-O notation, formal models of computation such as the lambda calculus and forged bridges between theoretical development and practical problem-solving applications;
5. The body of experience in proof checking and calculations involving logic was such that adapting it to work with general classical algebraic formulae could seem natural;
6. Initial planning or work was under way on many significant projects that will be described in the next section.
7. The concept of artificial intelligence inspired much work on carrying out procedures that humans find challenging, and working with algebraic formulae certainly fell into that category.

None of this work was isolated from wider social and cultural concerns. RAND Corporation, where IPL was developed, had spun off from a post-war US Air Force project to plan for future weapons. We view a verification of Russell and Whitehead as perhaps an unusual future weapon and celebrate the support that project was given.

[5] https://www.gamm.org/

The Soviet launch of Sputnik in October 1957 led President Eisenhower to establish the Advanced Research Projects Agency (ARPA) early the following year. One of its missions was to give the US the capacity to launch and use spacecraft. This called for work on automating calculations in celestial mechanics and orbital dynamics where pure numeric computation became strained and symbolic work was the best option. ARPA would go on to fund many of the projects we describe below.

3. 1965–1970

At the instigation of Jean Sammet, the ACM Special Interest *Committee* on Symbolic and Algebraic Manipulation (SICSAM) was formed in 1965 by George Forsythe, then President of the Association for Computing Machinery. In a 2006 oral history [11], Sammet recalled Forsythe writing back to her saying "I think that's a fine idea. You are now the Chairman..." Years later, Sammet recalled that one of the main motivations to form SICSAM was to have a forum where staff from different companies could share ideas [147]. Later this "Committee" became a "Group" and ACM SIGSAM was born. The following year, Sammet organized the first SYMSAC, which took place in Washington, DC and eventually led to the ISSAC conference series. "Everybody that went thought it was wonderful, and when you expected them to be out at the bar, they were in sessions!" she also recalled.

Although the ACM Digital Library does not record who was the General Chair of that conference, Sammet said she served as General Chair, Program Chair, and Chair of SICSAM, "a troika of one." Some of the papers were published in the *Communications of the ACM* after the conference. They include works on systems, on applications, on techniques for differentiation, and something that at first glance *looks* like it's for teaching—it's entitled "Grad Assistant"—but is not; the idea was that the program would replace your graduate student research assistant and do your drudge computations for you!
One more quotation, this one with an ironic outcome. Sammet said:

> There was a very strong Special Interest Group on numerical analysis [SIGNUM] in ACM, and at some point I think I contacted them, and, in a presumably polite way, they told me to go away and not bother them. They were interested in numerical analysis, and they didn't want any of this non-numerical stuff floating around. And that attitude lasted for a very, very long time.

When SIGNUM closed up shop around the turn of the 21st century, during the tenure of one of the present authors as Chair of SIGSAM or shortly thereafter, its remaining assets were assigned to SIGSAM. So the present ACM SIGSAM is the heir to both groups.

We mention one more item of interest from the archives of the SICSAM Bulletin (later the SIGSAM Bulletin, later *Communications in Computer Algebra*). The first Editor of the Bulletin was Peter Wegner[6]. He passed the torch after a year to John Young, who announced in his first message to the readership[7], that there would be a European sub-editor to whom correspondence should be addressed from that side of the Atlantic. That sub-editor was Professor Sir Maurice Vincent Wilkes, of Cambridge University. Wilkes won the Turing Award the next year, in 1967, and was knighted in 2000 for his many contributions to computer science. Early issues of the Bulletin also included a note on curriculum design for computer science by Donald E. Knuth (ACM Turing award 1974) co-authored with Peter Wegner. We will also note that "Tini" Veltman, whose 1963 algebra system Schoonschip [160] is another candidate for being "first," went on to become a Nobel Laureate. It seems that Sammet was right: there was significant interest in the community in symbolic and algebraic manipulation, and that attracted participants of extraordinary calibre.

Sammet identified the language Algy as the first published step towards a general-purpose language for computer algebra [13]. The authors were Myrna D. Bernick, E. D. Callender, and J. R. Sanford. The system Algy seemed to allow basic polynomial manipulation. Sammet credited this work with ideas that led to the creation of Formac in this period. Formac has been called the "first commercially successful computer algebra system." Elaine R. Bond's history of Formac, published in the *Proceedings of SYMSAC '66* acknowledged the influence of "other similar systems such

[6] Peter Wegner (1932–2017) was a professor of Computer Science at Brown University from 1969 to 1999. His work on object-oriented programming is considered seminal.
[7] On p. 1 of Issue #5

as ALPAK, ALTRAN and FORMULA ALGOL" and cited a paper of Sammet's that appeared in technical report form in 1965, and later in [144].

ALPAK (Algebra Package) was a set of SNOBOL routines and macros for symbolic manipulation of very large rational algebraic expressions. It was implemented by W. Stanley Brown and colleagues at Bell Labs in the early 1960s. This was succeeded by ALTRAN "a highly portable implementation of both algorithms and compiler... [which] included algorithmic advances in the handling of multivariate polynomials and macro generation to tailor FORTRAN code to make full use of the characteristics of particular hardware. Its features included a run-time environment with dynamic storage allocation, recursion, symbolic dumping, and error handling". One challenge raised by the ALTRAN team was finding multivariate greatest common divisors [20]. This led to intensive work in the early 1970s before being satisfactorily resolved [117].

It's part of the folklore of the symbolic computing community that 'the first program for symbolic differentiation was written before FORTRAN existed,' and this turns out to be true. In [10], a paper describing a system "MANIP" that could differentiate, we find citations that go back to 1962. That 1962 paper [84] cites two 1953 papers, one by Kharimanian and one by Nolan, and describes the process by which the programs had to be used: in each case the expressions had to be translated into a compact code before the programs could be run. These papers are credited in [145] as being the first.

Drawing further from [84], we infer several things. First, these early differentiation programs were seriously limited by the capacity of the computers used and a consequence was that the input and output of expressions had to be in really ugly restricted formats. The one in [84] accepted input in forms that looked like

```
A * X P 2 -I- B * X -4- SIN.(C * X)
```

They used "P" where today we might use "^". They parsed the input into a kind of Łukasiewicz prefix form [110] in a table and thought of this more as a table than as a tree.

The applications that they mention include a guided missile system, and they say "Equations consisting of three hundred or more symbols have been successfully differentiated with resultant derivatives of length in excess of seven hundred symbols. The equations under consideration involve six independent variables and four dependent variables."

If one can differentiate, then one can compute Taylor series. It turns out that for efficiency one should be careful, because naive methods lead to combinatorial growth in the length of symbolic expressions, as soon became painfully clear. So-called *automatic* differentiation emerged later to address this, but early work included [135]. That report described a program that generated FORTRAN and machine code for the particular computer in use at the author's institution in a way that we might classify as automatic differentiation today.

The papers so far highlighted here are concerned with system construction. Papers from applications also appeared in SYMSAC '66, such as [47], which described a FORMAC program to solve linear initial and boundary-value problems for ODE. The author, Elizabeth Cuthill (1923–2011), is perhaps most famous for her work on the *Cuthill–McKee* algorithm, which permutes sparse matrices in a way to reduce bandwidth. The paper describing that work is [48]. It's a judgement call to say that that work was "symbolic computation," but since it is so famous and so useful and has been cited thousands of times we are quite motivated to claim it for the community!

Another contribution in the applications line is [49] which described manipulation of Poisson series, that is, series where the terms are of the form

$$x_1^{k_1} x_2^{k_2} \cdots x_m^{k_m} e^{y_1} e^{y_2} \cdots e^{y_n} . \tag{1}$$

Such series[8] were, and are, in great demand for solving problems in celestial mechanics; see the famous paper [56].

James R. Slagle (1934-2023), who completed his 1961 dissertation under Marvin Minsky [152], described a LISP program called SAINT which could find antiderivatives of many functions. Slagle, who was blind, was also a chess champion. He won the first Over-the-Board tournament of the US Braille Chess Association.[9]

Joel Moses (1941–2022), who began his doctorate in 1963, redesigned Slagle's SAINT into a symbolic integration program called SIN, completed in 1967. It was also written in LISP and advanced the use of knowledge based systems rather than tree search for AI [122]. This work in LISP coincided with the creation of Project MAC at MIT (1963). Its first director was Robert M. Fano. Out of this project the MACSYMA system would be born.

[8] Or the more or less equivalent ones with sines and cosines in place of the complex exponentials

[9] https://web.archive.org/web/20160502160843/http://www.americanblindchess.org/potb.htm

MATHLAB was also a product of the mid-1960s, created at MITRE[10] by Carl Engelman (1929-1983) [64]. One of his rules stated "MATHLAB is intended for the physicist, not the programmer.", while another insisted:

> The computer, as viewed by the user, must be intimate and immediate. The user should have next to [his] desk a console consisting of a typewriter or, preferably, a typewriter and a scope. Economy might, in some cases, dictate the substitution of a plotter for the scope. These are connected to a large, fast, on-line, time-shared digital computer. [He] communicates with that computer by typing messages on his typewriter or by means of a light-pen on the scope. The computer replies by means of the same machines. It types both messages and equations. On the scope it displays both equations and graphs. Above all, the response time to the user's requests must be short.

It would be at least twenty to thirty years before that goal would be widely achieved. MATHLAB 2 introduced two-dimensional output (subscripts and superscripts in ASCII format).

SCRATCHPAD/1, written in LISP by James Griesmer, Dick Jenks and later David Yun [80], started in 1965 but was never publicly released. First versions are important for history, though: Griesmer's knowledge was later incorporated into SCRATCHPAD II by Dick Jenks and his team.

3.1. The gospel according to Moses

In about 2010, Joel Moses published his memoirs, entitled "My Life." These memoirs, which cover a much wider time period than we are restricting ourselves to in this paper, seem to have been published only online, at `https://www.csail.mit.edu/sites/default/files/2022-10/moses_memoirs.pdf`, and seem not to be widely cited (at least according to Google Scholar). We found these memoirs to be a significant source of information, particularly about the growth and influence of computer algebra in the United States but also about American academic culture of the time. We highly recommend reading them yourself, and we restrict consideration here to only a few points among many possible. We have entitled this section of our paper mostly as a friendly echo of Moses' sense of humour, which you may find examples of in his memoirs, but also as a reminder that in this section we are reporting his views, not necessarily our own.

Joel Moses obtained his PhD under the supervision of Marvin Minsky and Seymour Papert[11], and subsequently had several PhD students (sometimes co-supervised) who themselves had significant effects on our field and that of AI more generally, including Jamil Baddoura, Richard Fateman, Michael Genesereth, Barry Trager, Paul Wang, and David Yun. Moses published several influential papers, including [120, 121] and [119], "Algebraic simplification: A guide for the perplexed," which is possibly his most highly-cited paper.

Moses' classical and religious training coloured his views on artificial intelligence, in a way that may still be relevant today. He believed in the value of what he called "structured knowledge", meaning knowledge structured in layers, and he made a distinction between this and knowledge structured by use of hierarchical trees. His documentation of this belief, in his memoirs, includes tracing his family lineage back to the sixteenth century Rabbi of Prague, Rabbi Löwe, also called the Maharal, who is reputed to have created the Golem. Moses says that several of the researchers at MIT involved in AI in the early days also had been told family stories that they, too, were descended from this Rabbi, and that this had formed part of their motivation to study artificial intelligence. Indeed, somewhat whimsically, he dedicated his 1967 PhD thesis "To the descendants of the Maharal who are endeavoring to build a Golem."

On an even more whimsical note, we remark that the highly-cited simplification paper mentioned above was titled in clear echo of the 12th century work of Moses Maimonides, "The Guide for the Perplexed."

Moses claims that he was the one who chose the name MACSYMA:

> In the fall of 1968 Bill Martin, Carl Engelman and I began work on a project to combine our three prior efforts to create a mathematical system that would be the best of its kind. Bill led the effort to design the front-end and overall architecture of the new system. [...] I came up with the name of the new

[10] MITRE was a military think tank that was spun off from MIT Lincoln Labs in 1958. Its first employees were developers of the SAGE system which provided air defense during the Cold War.

[11] This present paper does not pursue the extremely important issue of the use of computer algebra or symbolic computation in education. Any paper that did so seriously would have to spend significant time on the work of Seymour Papert.

> system, Macsyma. Macsyma stood for Project MAC's SYmbolic MAnipulator. It also had an obvious Latin meaning, and a Hebrew one related to the Persian word "kismet" and meaning "magical" and "wondrous."

Phonetically, "MACSYMA" invokes the Latin "maxima," which is neuter plural substantive for "the greatest things". Google Translate simply says that "macsyma" in Latin means "maximum." It does not, apparently, know that it means magical or wondrous in Hebrew, although it does suggest "meksima" which it translates as "charming." Perhaps Google is perplexed.

Early in his career, Moses met Jean Sammet: indeed, she hired him to work on Project Mac. He had this to say about her:

> "Jean was a determined person. She would see the project to completion, and use it as a stepping-stone to increasingly greater roles in the computer society, ACM, the Association for Computing Machinery. She eventually became President of the ACM and a member of the National Academy of Engineering. One of her major achievements, in my view, was that she nearly single-handedly created the group of people who were interested in symbolic algorithms and systems, SIGSAM, the Special Interest Group for Symbolic and Algebraic Manipulation, becoming its first chairman. I became its third chairman a few years thereafter. IBM lent its considerable stature to the nascent field of symbolic computing with its support of FORMAC. Unfortunately FORMAC was a relatively poor product, but that gave the rest of the community of system builders a considerable advantage."

Moses volunteered to organize the second conference in computer algebra, in 1971, five years after the first one.

> "The symbolic algebra conference took place in July of 1971, with about two hundred in attendance–a good turnout for our field. I hurried from one session to another to make sure things were going well, rarely staying for a full session or even a full invited talk. I did make an exception for the talk on factorization of polynomials by Elwyn Berlekamp of Berkeley, who was something of a showman. He made it appear that he did not know what he was doing, and all the while he pulled technical rabbits out of the air. I loved his talk. In fact, the invited talks overall were of very high quality, as were most of the submitted papers."

Moses continued to work on integration, and on simplification, and on special functions in symbolic computation, throughout his non-administrative career. On the administrative side he rose to be Provost of MIT, and many pages of his memoirs deal with that, throwing around budgetary numbers that sound significant even today.

We draw attention to the following passage, on p. 221 in the section on the "Athena project" which was (apparently) the first serious attempt to use symbolic computation in education[12]:

> "Gerry also did not want us to do significant system development for Athena. I had hoped that UNIX would fill our needs, but I overestimated its capability in a large and complex environment such as Athena. We were saved by Jerry Saltzer, who became the technical director of the project. Under Jerry's supervision we introduced many improvements to UNIX that have become standards in the UNIX world. One was Kerberos, the software for authenticating users in the network. A key related addition was X-WINDOWS, initiated by a graduate student in LCS, and championed by a staff member from DEC, and approved by me for continuing development by Athena. X-WINDOWS permitted UNIX systems to emulate the windowing capability of a MAC computer. DEC felt that X-WINDOWS, which became DEC Windows in their environment, justified their entire investment in Athena."

One of the points of this present paper is to document places where computer algebra or symbolic computation has had an impact beyond the field. This passage reflects two important such instances.

There is a lot more to think about, as written in those memoirs. Moses did summarize some of this material for a later JSC paper, namely [122], but we believe that the richer memoirs are well worth reading, and we leave off here. You can also find an interview with him at the MIT infinite video archive.

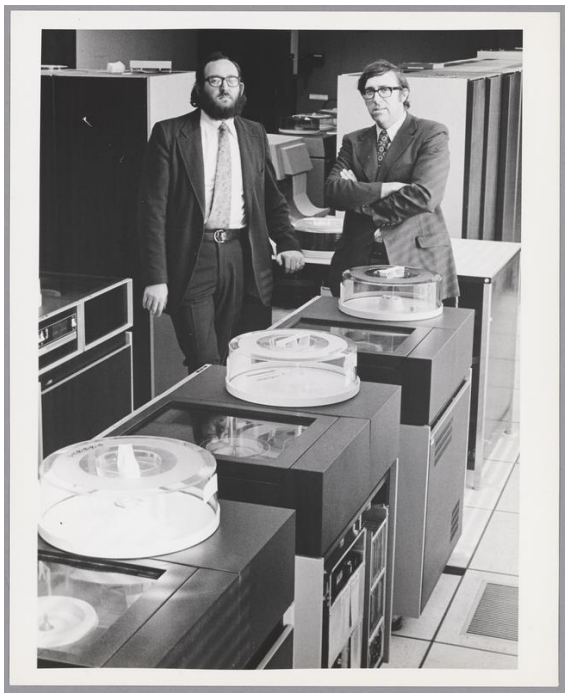
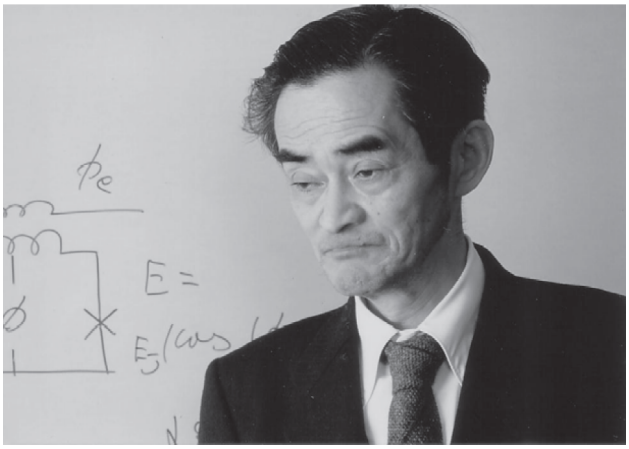


Figure 2: Joel Moses (left) and Michael Dertouzos (right) in the Laboratory for Computing at MIT in 1974. Photo (c) MIT 2010, courtesy MIT Museum.

(a) Hearn



(b) Goto

Figure 3: [left] Anthony C. Hearn. Photo by A. C. Norman. [right] Eiichi Goto. Photo taken from [92].

Figure 4: Bruno Buchberger. Photo courtesy Tomás Recio, taken on the occasion of Bruno's 80th birthday.

3.2. The world outside MIT

Reduce was created in the early-mid 1960s [88, 89] by Anthony C. Hearn. See Figure 3. The first published *use* of Reduce was [87], where it was reported that six months of human labour had been reduced to fifteen minutes of computer time. Reduce 2 appeared in 1970. One of the present authors has the full working source of that program, and its manual, from which we find that Reduce 2 did not have arbitrary-precision integers or bigfloats, and instead relied on whatever floating-point system was supplied by the underlying machine. Given the state of floating-point arithmetic in the years before the IEEE Standards, this seems natural. Reduce had to wait until 1979 for a bigfloat package [149]. Cobol, in contrast, already had facilities for specifying the number of figures before and after the decimal point fairly freely, which was another milestone for Jean Sammet. The first (and probably only) algebra system implemented in Cobol was described in 1976 [70]. To be fair, that system, while slow, actually had some quite interesting features.

These were not the only systems being contemplated at this time. John Cannon wrote a critique of algebraic programming languages in 1969, namely [32]. Cannon wrote that his "experience in developing programs to work with commutative and non-commutative polynomial rings" informed that critique, and doubtless formed a foundation for the development of Cayley, released in 1982 (and thus outside the scope of this paper) and later Magma. Then there was the language Formal (a "formula manipulation language") developed for NASA/JPL prior to 1970 by Charles Mesztenyi [115], who later went on to help develop GRAAL, a graph processing language.

Now, we must include a seminal contribution in the area of algebraic algorithms, namely the 1965 PhD thesis of Bruno Buchberger [21] which introduced Gröbner bases to computer algebra, and which was later translated to English in [26]. See Figure 4. The thesis was first followed by [22], which itself was translated to English in [25]. These major works and the Buchberger algorithm did not make their way into computer algebra systems until much later; in fact, not until after 1976 and the publication of [23]. Nevertheless these were important milestones for the era.

Before Gröbner bases, though, there was elimination theory using resultants. Significant prior work includes [83]. One very important paper of the late 1960s, now less well known than it deserves to be, is [100]. It made the claim that, for multivariate polynomials, Bézout matrices are superior to subresultants both in respect to speed and in respect to numerical stability. In view of the very pessimistic recent result [129] this may merit a second look with a modern lens.

Other important papers in polynomial arithmetic include Berlekamp's factorization of polynomials over finite fields [12] and Lipson's use of Chinese remaindering [109] for homomorphic methods.

A fundamental and practical advance was made in [63] and independently in [4], where an efficient method for solving linear systems of equations over the integers was given. This was the first of the "fraction-free" methods.

[12] As we have stated, we are not pursuing education in this paper; we hope to document this aspect in future work.

Later, in [5], this was extended to general integral domains. The first paper, by Jack Edmonds, which already gives its result for general integral domains, apparently gives a counterexample to what is known as "The Matthew Effect" wherein the more famous scientist is given most of the credit; here, although Bareiss is arguably less famous than Edmonds (who, for instance, won the inaugural von Neumann Theory Prize in 1985), the algorithm is widely known as Bareiss' algorithm. This seems to be true in spite of the fact that Edmonds' paper has 512 citations in Google Scholar at this time of writing, nearly the same as Bareiss' paper which has 538. Bareiss did publish an internal report on this in May 1966 for Argonne National Lab [3], while Edmonds' published work is dated November 1966 on the paper itself, although the publication date of the journal seems to be 1967. However, according to the Matthew effect (a play on the Gospel theme from Matthew 13:12 "For whosoever hath, to him shall be given, and he shall have more abundance: but whosoever hath not, from him shall be taken away even that he hath."), Edmonds should have been able to acquire the lion's share of the credit anyway. This, of course, is not to be confused with "Stigler's Law" which says that the name given to any concept is never that of the scientist who originally thought of it. Michael Monagan of Simon Fraser University informs us that he is now calling it Bareiss/Edmonds fraction-free Gaussian elimination. The Wikipedia page for the "Bareiss Algorithm," as it is currently known, gives a link for a crisp scan of the book [3], which might otherwise be hard to find.

A huge development in the theory of integration in finite terms was published in this time period, namely the papers of Robert H. Risch giving (in outline) an algorithm to either find an antiderivative of a given elementary function, or to prove that no such expression was possible. See [137], [138], and [142] for a more accessible introduction. See also [141], which seems to have been the paper which converted the earlier analytic techniques of Liouville and of Ritt into the algebraic terms that we know now. Implementation of this algorithm took quite some time and effort; the *Proceedings of EUROSAM '79* contain four important papers by Norman, Davenport, Trager, Moses and Zippel which helped to bring the development to a more satisfactory state. Of course, research continues today.

At the risk of venturing too far outside of the symbolic computation world, we mention the pioneering work of Stephen A. Cook on the complexity of computation of functions, beginning with multiplication, for example the 1969 paper [45]. This was soon followed by fast multiplication by Schönhage and Strassen [150]. The paper by Strassen [155] was a fundamental advance in improved complexity for linear algebra. The field of "computational complexity" has had a profound effect on research in computer algebra ever since. See, for instance, the textbook [130].

We now mention the 1969 undecidability result of Richardson [136], and the 1970 paper of Caviness [33]. According to Joel Moses [122], Richardson's result was a bit controversial at the time because of his inclusion of "absolute value" in the collection of functions allowed in the expression to be processed. Nonetheless, the fact that recognizing zero is undecidable over such a simple class of expressions remains remarkable. Specifically, if the expression E contains $\ln 2$, π, $\exp(x)$, and $\sin(x)$, then the problem of determining if $E < 0$ is undecidable. Adding the function $|x|$ to the list of possibilities makes determining if $E = 0$ undecidable. As discussed in [33] this result has many implications for practical symbolic computation, because simplification is fundamental. Because (some) simplification problems are undecidable, we must sometimes rely on heuristics, and this leads to a forest of difficulties even today.

4. 1971–1975

The very efficient and compact Camal (for Cambridge Algebra Language), by David Barton, Stephen Bourne[13], John Fitch and others, started its gestation in about 1968 according to the delightful little history [69] (see also [6, 7]) but was not published until 1971.

Macsyma became publicly available in 1971, after developments mentioned in the previous section. The history of Macsyma is sketched broadly in [57], from the point of view of a historian of computing. The comments there might be fascinating to the insider: we see that in addition to providing tools for human use, Macsyma forced the humans to adapt to the tools. HAKMEM (a February 1972 AI Lab memo) also came out of MIT around this time. Compiled by Michael Beeler, R. William Gosper, and Richard C. Schroeppel, it described 'little known data' of interest to computer hackers, 'to save some duplication of effort–except for fun' [9]. It has become known to subsequent generations

[13] Stephen R. Bourne is known for the Bourne shell, for work on Algol68, and for having been President of the ACM. See his Wikipedia page https://en.wikipedia.org/wiki/Stephen_R._Bourne.

through references like [107] and [165]. MACSYMA was weighted more heavily towards the knowledge of special functions than were the other systems at the time, and this memo reflects an important emphasis of that software system.

ALTRAN continued its development during this time. We note that Morven Gentleman (1942–2018) is credited in the fourth edition of the ALTRAN manual as being one of the developers, probably in the early 1970s; some of us knew him later as an advisor for the creation of the 1980s program MAPLE. Gentleman published a paper in the SIGSAM Bulletin using ALTRAN for Truncated Power Series [73]. Another related result was Gentleman showing that, unexpectedly at the time, the optimal multiplication chain for powering a polynomial was repeated multiplication by the original polynomial [72].

Also in this period, George Collins (1928–2017) introduced his SAC-1 system for polynomial arithmetic [40]. This system, written in FORTRAN, included functions for polynomial GCD, factorization, resultants, exact real zero calculation, partial fraction decomposition, rational function integration, and solution of systems of linear equations with polynomial coefficients.

4.1. Tony Hearn and REDUCE

There was a serious change in Cambridge brought about through the agency of Tony Hearn. The ARPAnet was very young and a transatlantic link joined up to London. In Cambridge and with the excuse of a new multi-way collaboration that Hearn was setting up (him in Utah, but Goto's group in Tokyo, the Cambridge few and David Stoutemyer in Hawaii) Cambridge got a 110 baud teletype with a modem which and could dial the London node and connect across to the Utah machine.

Quoting from https://www.sciencemuseum.org.uk under objects-and-stories and arpanet-internet#what-was-the-arpanet:

> "In the UK, the National Physical Laboratory (NPL) created its own packet switching Local Area Network in 1971.
>
> Then, on 25 July 1973, a computer at University College London (UCL) sent packets of digital information to another in California. Led by Peter Kirstein at the Institute for Computer Science, UCL, this was ARPANET's first transatlantic link.
>
> It ran from London to Norway via cable, then on to the Seismic Data Analysis Centre (SDAC) in Virginia via satellite link, and finally across the US network to the Information Sciences Institute at the University of Southern California.
>
> The ARPANET team had originally identified NPL as its first London link, but in the context of developing relationships within Europe it was deemed politically unacceptable for a British government research laboratory to connect directly with the US Department of Defense."

At that time the ARPANET had around 2 dozen nodes.

Involvement in that project helped justify grant funding at Cambridge that got Mary-Ann Moore and Arthur Norman visiting David Stoutemyer in Hawaii to work on 2D printing on a fairly dumb terminal for the algebra code he had at the time[14], as well as a multivariate polynomial GCD based on a subtractive remainder sequence. Norman went on to Tokyo which was when he first met Kanada, Sasaki, and other members of Goto's group. Subsequently, several Japanese researchers arranged to visit Cambridge for various non-trivial amounts of time.

Tony Hearn visited Cambridge for a summer at a stage that John Fitch and Arthur Norman were just getting CAMBRIDGE LISP working well enough to be able to try to mount REDUCE on it. Arthur Norman recalls an anecdote from that episode, which is that an attempt crashed and after some digging it turned out that a function that had been compiled into machine code had just been redefined. A consequence was that the code involved happened to be being executed but the reference that kept it alive was replaced with a pointer to the new version—and his code garbage collected compiled code chunks as well as simple data. So the only reference to this code was a return address into it on the call stack. When the return happened, the code had gone, and the memory was used for other things. So he

[14] This 2D printing also had an elegant line-breaking algorithm which made use of labels for subexpressions.

needed to adjust the garbage collector so that it found return addresses and tagged code blocks they pointed into as 'alive'.

REDUCE users were early adopters of international networking, and that over time this grew so that REDUCE had contributions from many parts of the world from China and Japan to the West of Utah right out to Russian scientists to the East, with Tony Hearn running a sort of shuttle diplomacy visiting sites where he could assist both physicists working out how to solve their problems and potential or actual contributors of packages that added capabilities. The portability of his code and the very open way in which it was circulated built a community that is perhaps best seen as foreshadowing the open source models commonplace today. This community was very much not centralised at one particular institution.

This growth of community was somewhat different to that of MACSYMA which was more centralised at project MAC and probably mostly made available by remote access to the project MAC computers[15]. In contrast REDUCE was, already at the time, running on multiple different and sometimes eccentric computers—indeed FLATS was purposely being built with REDUCE in mind—and this was happening globally, in groups each of which was individually probably fairly small.

4.2. Scratchpad/1

It is hard to overestimate the influence of SCRATCHPAD II project in the 1980s that followed SCRATCHPAD/1 from the 1970s. The heady "pure research" environment of IBM T. J. Watson Research Center provided a hub for the exchange of ideas and technology. The interchanges between visiting researchers and all kinds of scientists and engineers led to many downstream effects. The principal feature that emerged from SCRATCHPAD II and that differentiated it from other computer algebra systems (before or since) was its formalization of the domains of computation. This enabled both rigour and efficiency in a way that was both challenging (to the user, sometimes) and satisfying to the researcher.

An example paper from the 1970s is [133]. This paper appeared in ACM *Transactions on Mathematical Software* (TOMS), which even then was an excellent vehicle for publication[16]. A comparison to the report [135] published ten years earlier shows considerable development: we see for instance what is now called "lazy evaluation," as well as considerably improved ease of use. Portability, however, remained an issue for both works, although for completely different reasons.

4.3. Algebraic algorithms

Leaving systems for the moment and considering algorithms, major progress was made in this time period on the important problems of multivariate GCD [123] and factorization [163, 164] using ideal-adic methods [170]. This work—building on the work of others such as Hans Zassenhaus [171] as it does—marks the take-off of sophisticated modular methods in computer algebra systems, which remain crucial system tools for efficiency. This was followed up by an influential Mathematics of Computation paper by Paul Wang and Linda Preiss Rothschild [164], again showing increasing impact of symbolic computing on the wider community.

Several people have since observed that, *in applications*, multivariate polynomials factor frequently and usefully. This implies that the results of these fundamental papers have had, in all likelihood, an outstanding if underappreciated effect on the utility of symbolic computation. Since multivariate polynomials with coefficients "chosen at random" mostly do not factor at all, perhaps the extreme utility of this work could not have been anticipated.

4.4. The beginning of theorem-provers

Perhaps the most highly-cited work ever published in the SIGSAM Bulletin was George E. Collins' follow-up to [41], namely the abstract [42] which is just outside our time frame, although the first one is inside our time frame

[15] There is some discussion of this in Moses' memoirs, who mentions the early role of Richard Fateman at Berkeley in encouraging dispersal of MACSYMA.

[16] Some of the present authors feel that this journal was not as well-used historically by the symbolic computation community as it might have been, perhaps because it was viewed as primarily a "numerical" software journal. This (very symbolic) paper shows that that perception was incorrect. There were a few further symbolic computation papers in TOMS since this paper, and they tended to have good impact outside our community, but there were not that many. More might have been better.

and is the one that has the actual results (the second is just an abstract[17]). As of January 2025 that abstract had been cited more than two and a half thousand times. This reflects the importance of the topics of cylindrical algebraic decomposition and quantifier elimination. Collins cites Tarski [156] in that first-mentioned paper for the foundational work.

4.5. Continued growth of applications

Descriptions of other applications of computer algebra continued to be published in this time period. For instance, consider [8], a very substantial paper of over seventy pages (including a seven-page bibliography). This paper surveyed various applications in physics: celestial mechanics, general relativity, and quantum electrodynamics. It also gave an overview of computer algebra systems for physicists (not just REDUCE), and from this paper[18] we learn that SIN by Joel Moses (mentioned above) already used Hermite reduction to perform integration by partial fractions.

5. Towards more modern times

We leave off discussion of the history of symbolic computation with a few short remarks leading to the present day. We do not wish to give the impression that we believe that the current state is uniformly better than that of the past. Indeed we do *not* hold that opinion, and instead lament some lost opportunities (for efficiency, for example). But there is no question that the fifty years since 1975 have wrought great change and in many cases very significant advances. We do not tell that story here even in the laconic style we used for the years 1965–1975, but rather just mention a few items.

- The first is the 1982 book edited by Buchberger, Collins, Loos, and Albrecht. The table of contents is printed in [27], where we see that a comprehensive overview of the field is attempted for the first time. A significant collection of historical material is present in the volume itself. This clearly marks a degree of maturity in the field;
- The first textbooks on computer algebra (it may be invidious to mention any particular one, but for instance the book [50] was translated to many languages);
- Wen-tsün Wu[19] and the method of characteristic sets 1978 [71], starting the topic of automated deduction in geometry by computer algebra tools that is still quite active (ADG conference series, see `https://adg-foundation.info`, `https://adg-foundation.info/conferences.html`);
- The founding of the *Journal of Symbolic Computation* in 1985;
- The founding of the Research Institute for Symbolic Computation (RISC)-Linz in 1987, of the Key Laboratory of Mathematics-Mechanization (KLMM)-Beijing, also in 1987, and of the Ontario Research Centre for Computer Algebra (ORCCA)-London, Ontario, in 1997;
- The rise of the most popular modern commercial systems, namely MAPLE and MATHEMATICA, and the MATLAB SYMBOLIC TOOLBOX (initially using MAPLE).
- The creation and development of CAYLEY, later MAGMA, MACAULAY, CoCoA and SINGULAR. These last three were all started in the 1980s and specialized in computational algebraic geometry;
- The D5 principle, as expounded by Dominique Duval and others [59, 55];

[17] At times one gets the idea that citation culture is irrational. We use Google Scholar to count citations because of its wide disciplinary coverage. Although flawed, Google Scholar citation counts are just a proxy for impact anyway, and because we use this consistently at least it's a reasonably fair comparator. Being off by, say, ten percent is entirely likely, but with this many citations that doesn't matter.

[18] We might have remembered this from [120], of course.

[19] In the Western manner of family name last, his name is spelled variously Wenjun Wu, Wen-Tsun Wu, Wen-tsun Wu, and Wen-tsün Wu. He himself used the latter, and so we do the same.

- The merging of the SYMSAC conference series [148, 134, 95, 161, 37] (held every 5 years from 1966 to 1986) with the interleaved European EUROSAM/EUROCAM/EUROCAL stream [94, 128, 31, 132, 68, 24, 34, 51] to form the named ISSAC stream in 1988 [74, 76, 166, 167, 162, 18, 111, 106, 102, 101, 75, 61, 158, 124, 118, 151, 82, 97, 62, 19, 93, 113, 169, 108, 131, 98, 125, 139, 140, 30, 2, 16, 112, 116, 85, 58, 86, 103], together annual since 1981 (for conference chairs and program chairs see [43]);
- Developing visions of the future [15, 168];
- Macsyma, Axiom and Reduce, all of which had previously been either commercial or otherwise subject to redistribution constraints, were re-licensed as open source, and the general purpose capabilities they offered thus became available to all cost-free. This also means that the algorithms and techniques embedded within them could be studied by anybody who was interested, and various volunteers support them right up to the current day. These will now count as among the oldest software products still being maintained and used;
- Continuing development of applications (for instance, the Canadarm[20])
- Continuing development of practical methods for exact solution of linear systems [60];
- Portmanteau mathematical, engineering and educational support software, led by SageMath where symbolic computation is supported alongside numerical work, visualization and document preparation and presentation;
- Hardware design motivated by the special needs of our subject, including both the Xerox D-series machines, the MIT Lisp machines[21] [79] and FLATS[77]. Some of these hardware projects have left a lasting impact on modern ways of using personal computers, while others were eclipsed by the rising tide of cheap fast processors. See the obituary of Eiichi Goto (1931–2005) [92].

5.1. Other histories

We have mentioned already some historically-oriented papers. Then there is the telegraphic list at Brian Evans's History of Computer Algebra[22].

In [159] we find a history of computer algebra starting in the 1980s, traced back through the Spanish engineer and inventor Torres–Quevedo (1852–1936) and before. That paper takes as its actual starting point the results of SYMSAC '86 and the meeting of the one of the present authors there with Wen-tsün Wu, and works its way backward.

Many other papers attempt at least a brief summary of the history of computer algebra. One notable such paper is [54], which begins with Babbage and notes several milestones.

6. Significant omissions

We have not discussed the history of the use of computer algebra in education, even though it has been very important for many members of our community since the beginning. We might have included the creation of the computer language Logo in 1967. Instead we point to the history at [153], and leave that story for another day.

Automatic reasoning, then called theorem proving, was mentioned as an interest of the members of SICSAM in the very first issue of the Bulletin. Yet we have not mentioned much about the developments in 1965–1975 apart from the quantifier elimination work. There is much more to say.

We mentioned the two-dimensional math output of MathLab, but the whole subject of human-computer interaction has had a "test laboratory" in computer algebra systems. Input of complex objects, control of highly structured objects where transformations from one class to another are routinely desired, and sophisticated visualization of the output have all been desiderata. Simultaneously there has been a strong need for standardized dissemination of the results.

[20] https://www.asc-csa.gc.ca/eng/canadarm/, http://www.maplesoft.com/view.aspx?SF=141144/ChangingFaceRobotic.pdf

[21] See https://en.wikipedia.org/wiki/Lisp_machine. See also the discussion at https://en.wikipedia.org/wiki/Symbolics and the adversarial connection to the modern GNU project.

[22] http://felix.unife.it/Root/d-Mathematics/d-The-mathematician/d-History-of-mathematics/t-History-of-computer-algebra

Neither have we talked about visualization or illustration of mathematical concepts. One very amusing visualization in the 1975 Bulletin is by Donald E. Knuth [99], where in describing the then-current state of his "Volume 2, Seminumerical algorithms" he includes a hand-drawn chart of editing changes.

7. Concluding remarks

We have given a brief summary of work in computer algebra and symbolic computation, concentrating on the period 1965–1975, ending fifty years ago. This time period was chosen in part because ISSAC 2025 in Guanajuato, Mexico was the fiftieth since SYMSAC '66, the founding conference organized by Jean Sammet. We have spent a bit more space in this paper in detailing her contributions, which we believe were formative for the community.

During 1965–1975, several important features of our current set of tools were established: efficient methods for dealing with polynomials and series, effective methods for differentiation and integration, methods for solving linear and nonlinear algebraic equations, and manipulations of special functions. One major idea driving our subject early on was Artificial Intelligence. Another could loosely be termed "solving equations". A major motivating application was computing formulas in a variety of areas of physics. In each case the work was notable for being sharply constrained by the amount of memory that was available.

A half-century and more later, we can see the discipline of "symbolic computation," which had split off into its own island state for a while, beginning to move back to merge with other sorts of computer work, bringing with it its style of mathematical discipline. And as it impinges on AI it joins in where hundreds of gigabytes strain current resources. Application areas that will have benefited from analytic derivation of the exact computation rules for computational chemistry and fluid dynamics and others are in a similar situation.

Jean Sammet observed early on that "expressions grow to exceed the space available" [145]. This has held true to a really astonishing extent! Even today, Gröbner Bases and Quantifier Elimination can have explosive memory demands even on problems of modest size.

It is becoming evident that knowing the history of computer algebra and symbolic computation is important for the general public, not just for us. In the week previous to ISSAC 2025, some of the authors of this paper ran a session on historical topics at the 30th Applications of Computer Algebra (ACA) conference in Crete. The plan there was to have a number of first-hand reports from individuals who were present in earlier days, and then to form a group to build up a fuller record of this history in book form. That session was a success, and the project is ongoing. Those who read this paper and who have relevant information or insight are encouraged to get in touch so that they can provide input to that enterprise.

Acknowledgements

This project was begun in a conversation at a meeting of the Computational Epistemology Think Tank run by The Rotman Institute of Philosophy. We have relied significantly on the record kept by the ACM Digital Library (it is somewhat ironic, given that this is a paper on the early history of artificial intelligence, that we had to certify so many times when accessing the Library that we were human). We are grateful to Bruno Buchberger for encouragement and comments on an earlier draft. We also thank Michael Monagan, David Stoutemyer, and the ISSAC referees for comments on the previous version of the paper [46], for all their extremely helpful comments and suggestions. Thanks also to Santiago Recio, for his help dealing with the possible Latin inspiration of the name MACSYMA.

References

[1] Janet Abbate. *Jean Sammet, an oral history*. IEEE History Center, Piscataway, NJ, USA, 2001.

[2] Carlos Arreche, editor. *ISSAC '18: Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2018. ACM.

[3] Erwin H. Bareiss. *Multistep integer-preserving Gaussian elimination*, volume 7213. Argonne National Laboratory, May 1966.

[4] Erwin H. Bareiss. Sylvester's identity and multistep integer-preserving gaussian elimination. *Mathematics of computation*, 22(103):565–578, 1968.

[5] Erwin H. Bareiss. Computational solutions of matrix problems over an integral domain. *IMA Journal of Applied Mathematics*, 10(1):68–104, 1972.

[6] David Barton. A scheme for manipulative algebra on a computer. *The Computer Journal*, 9(4):340–344, 02 1967.

[7] David Barton, Stephen R. Bourne, and Colin J. Burgess. A simple algebra system. *The Computer Journal*, 11(3):293–298, 01 1968.

[8] David Barton and John P. Fitch. Applications of algebraic manipulation programs in physics. *Reports on Progress in Physics*, 35(1):235, 1972.

[9] Michael Beeler, R. William Gosper, and Richard Schroeppel. HAKMEM. Technical Report AIM 239, MIT AI Lab, Feb 1972.

[10] Bernice Bender. A computer system for algebra and analytic differentiation. In *Proceedings of the First ACM Symposium on Symbolic and Algebraic Manipulation*, SYMSAC '66, page 0201–0227, New York, NY, USA, 1966. ACM.

[11] Thomas J. (Tim) Bergin and Jean E. Sammet. Jean E. Sammet interview: March 28, April 4, April 11 and April 18, 2006. In *ACM Oral History Interviews*, New York, NY, USA, 2006. ACM.

[12] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46:1853–1859, 1967.

[13] Myrna D Bernick, E. David Callender, and J. R. Sanford. ALGY—an algebraic manipulation program. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 389–392, 1961.

[14] Jonathan P Bowen. A brief history of algebra and computing: An eclectic Oxonian view. *IMA Bulletin*, 31(1/2):6–9, 1995.

[15] Ann Boyle and Bobby F. Caviness. Future directions for research in symbolic computation. In *Report on Symbolic Algebraic Computations Workshop, April, TR-200*, pages 29–30, Washington DC, USA, 1988. Society for Industrial and Applied Mathematics.

[16] Russel Bradford, editor. *ISSAC '19: Proceedings of the 2019 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2019. ACM.

[17] Claude Brezinski, Gérard Meurant, and Michela Redivo-Zaglia. *A Journey through the History of Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2022.

[18] Manuel Bronstein, editor. *ISSAC '93: Proceedings of the 1993 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1993. ACM.

[19] C.W. Brown, editor. *ISSAC '07: Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2007. ACM.

[20] W. S. Brown. A language and system for symbolic algebra on a digital computer. In *Proceedings of the first ACM symposium on Symbolic and algebraic manipulation - SYMSAC '66*, SYMSAC '66, page 501–540. ACM Press, 1966.

[21] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, Mathematical Institute, Leopold Franzens University, Innsbruck, Austria, 1965.

[22] Bruno Buchberger. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes Mathematicae*, 4(3):374–383, October 1970.

[23] Bruno Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.*, 10(3):19–29, August 1976.

[24] Bruno Buchberger, editor. *EUROCAL '85: European Conference on Computer Algebra, Volume 1: Invited Lectures*, number 203 in LNCS, Berlin, Heidelberg, 1985. Springer Verlag.

[25] Bruno Buchberger. An algorithmic criterion for the solvability of algebraic systems of equations. In Bruno Buchberger and Franz Winkler, editors, *Gröbner Bases and Applications*, volume 251 of *London Mathematical Society Lecture Note*, pages 533–545. Cambridge University Press, 1998.

[26] Bruno Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3):475–511, 2006. Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday).

[27] Bruno Buchberger, George E. Collins, Rüdiger Loos, and Rudolf Albrecht. Computer algebra symbolic and algebraic computation. *SIGSAM Bull.*, 16(4):5, November 1982.

[28] Bruno Buchberger, George E. Collins, and Rüdiger Loos with the cooperation of Rudolf Albrecht, editors. *Computer algebra — symbolic and algebraic computation*. Computing Supplementa 4. Springer, 1982.

[29] Bruno Buchberger et al. Symbolic computation (an editorial). *Journal of Symbolic Computation*, 1(1):1–6, 1985.

[30] Michael Burr, editor. *ISSAC '17: Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2017. ACM.

[31] Jacques Calmet, editor. *EUROCAM '82: European Computer Algebra Conference*, number 144 in LNCS, Berlin, Heidelberg, 1982. Springer Verlag.

[32] John J. Cannon. A critique of algebraic programming languages. *SIGSAM Bull.*, 12:18–27, Jan. 1969.

[33] Bobby F. Caviness. On canonical forms and simplification. *Journal of the ACM (JACM)*, 17(2):385–396, 1970.

[34] Bobby F. Caviness, editor. *EUROCAL '85: European Conference on Computer Algebra, Volume 2: Research Contributions*, number 204 in LNCS, Berlin, Heidelberg, 1985. Springer Verlag.

[35] Bobby F. Caviness and George E. Collins. Symbolic mathematical computation in a ph. d. computer science program. *SIGCSE Bull.*, 4(1):19–23, March 1972.

[36] Bobby F. Caviness and George E. Collins. Symbolic mathematical computation in a ph.d. computer science program. *SIGSAM Bull.*, 23:25–28, Jul. 1972.

[37] Bruce W. Char, editor. *SYMSAC '86: Proceedings of the fifth ACM symposium on Symbolic and algebraic computation*, New York, NY, USA, 1986. ACM.

[38] Alonzo Church. Correction to A note on the Entscheidungsproblem. *The Journal of Symbolic Logic*, 1(3):101–102, 1936.

[39] Alonzo Church. A note on the Entscheidungsproblem. *The Journal of Symbolic Logic*, 1(1):40–41, 1936.

[40] George E. Collins. The SAC-1 system: An introduction and survey. In *SYMSAC'71: Proceedings of the second ACM Symposium on Symbolic and algebraic manipulation*, pages 144–152. ACM, 1971.

[41] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition–preliminary report. *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, 8(3):80–90, August 1974.

[42] George E Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition: a synopsis. *ACM SIGSAM Bulletin*, 10(1):10–12, 1976.

[43] ISSAC Steering Committee. Past ISSAC conferences. https://www.issac-conference.org/past.php, 2025.

[44] Stephen A. Cook. *On the minimum computation time of functions*. PhD thesis, Harvard University, 1966.

[45] Stephen A. Cook and Stál O. Aanderaa. On the minimum computation time of functions. *Transactions of the American Mathematical Society*, 142(0):291–314, 1969.

[46] Robert M. Corless, Arthur C. Norman, Tomas Recio, William J. Turkel, and Stephen M. Watt. Symbolic mathematical computation 1965–1975: The view from a half-century perspective. In *Proceedings of the 2025 International Symposium on Symbolic and Algebraic Computation*, ISSAC '25, page 140–149, New York, NY, USA, 2025. Association for Computing Machinery.

[47] Elizabeth Cuthill. A FORMAC program for the solution of linear boundary and initial value problems. In *Proceedings of the First ACM Symposium on Symbolic and Algebraic Manipulation*, SYMSAC '66, page 0801–0850, New York, NY, USA, 1966. ACM.

[48] Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, page 157–172, New York, NY, USA, 1969. ACM.

[49] J. M. A. Danby, André Deprit, and A. R. M. Rom. The symbolic manipulation of Poisson series. In *Proceedings of the First ACM Symposium on Symbolic and Algebraic Manipulation*, SYMSAC '66, page 0901–0934, New York, NY, USA, 1966. ACM.

[50] J. H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra, Systems and Algorithms for Algebraic Computation*. Academic Press, 2nd edition, 1993.

[51] James H. Davenport, editor. *EUROCAL '87: European Conference on Computer Algebra*, number 378 in LNCS, Berlin, Heidelberg, 1987. Springer Verlag.

[52] Amanda Davis. Jean Sammet, the accidental programmer. *IEEE Spectrum*, December 2024.

[53] Martin Davis, Hilary Putnam, and Julia Robinson. The decision problem for exponential Diophantine equations. *The Annals of Mathematics*, 74(3):425–436, November 1961.

[54] Wolfram Decker. Some introductory remarks on computer algebra. In *European Congress of Mathematics: Barcelona, July 10–14, 2000 Volume II*, pages 121–142. Springer, 2001.

[55] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. *About a new method for computing in algebraic number fields*, volume 204 of *Lecture Notes in Computer Science*, pages 289–290. Springer Berlin Heidelberg, 1985.

[56] André Deprit. Canonical transformations depending on a small parameter. *Celestial mechanics*, 1(1):12–30, 1969.

[57] Stephanie A. Dick. Coded conduct: making MACSYMA users and the automation of mathematics. *BJHS Themes*, 5:205–224, 2020.

[58] Alicia Dickenstein, Elias Tsigaridas, and Gabriela Jeronimo, editors. *ISSAC '23: Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2023. ACM.

[59] Claire Dicrescenzo and Dominique Duval. *Algebraic extensions and algebraic closure in Scratchpad II*, volume 358 of *Lecture Notes in Computer Science*, page 440–446. Springer Berlin Heidelberg, 1989.

[60] John D. Dixon. Exact solution of linear equations using p-adic expansions. *Numerische Mathematik*, 40(1):137–141, 1982.

[61] Sam Dooley, editor. *ISSAC '99: Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1999. ACM.

[62] Jean-Guillaume Dumas, editor. *ISSAC '06: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2006. ACM.

[63] Jack Edmonds. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards Sect. B*, 71(4):241–245, 1967.

[64] Carl Engelman. MATHLAB: a program for on-line machine assistance in symbolic computations. In *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part II: computers: their impact on society*, pages 117–126, 1965.

[65] D.J. Farber, R.E. Grisold, and I.P. Polonsky. A preliminary report on the string manipulation language SNOBOL. Technical Report Unpublished Technical Memorandum 63-3344-2, Bell Telephone Laboratories, 1963.

[66] D.J. Farber, R.E. Grisold, and I.P. Polonsky. SNOBOL, a string manipulation language. *Journal of the ACM*, 11(1):21–30, 1964.

[67] Lawrence M Fisher. Jean E. Sammet 1928–2017. *Communications of the ACM*, 60(7):22–22, 2017.

[68] John P. Fitch, editor. *EUROSAM 84: International Symposium on Symbolic and Algebraic Computation*, number 174 in LNCS, Berlin, Heidelberg, 1984. Springer Verlag.

[69] John P. Fitch. CAMAL 40 years on - is small still beautiful? In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Intelligent Computer Mathematics, 16th Symposium, Calculemus 2009, 8th International Conference, MKM 2009, Held as Part of CICM 2009, Grand Bend, Canada, July 6-12, 2009. Proceedings*, volume 5625 of *Lecture Notes in Computer Science*, pages 32–44. Springer, 2009.

[70] John P. Fitch, P. Herbert, and Arthur C. Norman. Design features of COBALG. In *Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 185–188, 1976.

[71] Xiao-Shan Gao. Wen-Tsun Wu: His life and legacy. *ACM Commun. Comput. Algebra*, 51(2):73–79, October 2017.

[72] W. Morven Gentleman. Optimal multiplication chains for computing a power of a symbolic polynomial. *Mathematics of Computation*, 26(120):935–939, 1972.

[73] W. Morven Gentleman. Experience with truncated power series. *SIGSAM Bull.*, 8(3):61–62, 1974.

[74] P. Gianni, editor. *Symbolic and Algebraic Computation: International Symposium ISSAC 88*, number 358 in LNCS, New York, NY, USA, 1988. Springer Verlag.

[75] Oliver Gloor, editor. *ISSAC '98: Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1998. ACM.

[76] G.H. Gonnet, editor. *ISSAC '89: Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1989. ACM.

[77] E. Goto, T. Soma, N. Inada, T. Ida, M. Idesawa, K. Hiraki, M. Suzuki, K. Shimizu, and B. Philipov. Design of a Lisp machine - FLATS. In *Proceedings of the 1982 ACM Symposium on LISP and Functional Programming*, LFP '82, page 208–215, New York, NY, USA, 1982. ACM.

[78] Johannes Grabmeier, Erich Kaltofen, and Volker Weispfenning, editors. *Computer algebra handbook*. Springer, Berlin, Germany, 2003 edition, November 2002.

[79] Richard D. Greenblatt, Thomas F. Knight, John T. Holloway, and David A. Moon. A LISP machine. *SIGIR Forum*, 15(2):137–138, March 1980.

[80] James H. Griesmer and Richard D. Jenks. SCRATCHPAD/1—an interactive facility for symbolic mathematics. In *Proc. 2nd Symposium on Symbolic and Algebraic Manipulation*, pages 45–53. ACM, 1971.

[81] Ralph E. Griswold. A history of the SNOBOL programming langauges. *ACM Sigplan Notices*, 13(8), 1978.

[82] Jaime Gutierrez, editor. *ISSAC '04: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2004. ACM.

[83] Walter Habicht. Zur inhomogenen Eliminationstheorie. *Commentarii Mathematici Helvetici*, 21(1):79–98, December 1948.

[84] James W. Hanson, Jane Shearin Caviness, and Camilla Joseph. Analytic differentiation by computer. *Commun. ACM*, 5(6):349–355, June 1962.

[85] Amir Hashemi, editor. *ISSAC '22: Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2022. ACM.

[86] Jonathan Hauenstein, Wen-shin Lee, and Shaoshi Chen, editors. *ISSAC '24: Proceedings of the 2024 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2024. ACM.

[87] Anthony C. Hearn. Computation of algebraic properties of elementary particle reactions using a digital computer. *Comm. ACM*, 9(8):573–577, 1966.

[88] Anthony C. Hearn. REDUCE: A user-oriented interactive system for algebraic simplification. In M. Klerer and J. Reinfelds, editors, *Interactive Systems for Experimental Applied Mathematics*, pages 79–90, New York, 1968. Academic Press.

[89] Anthony C. Hearn. REDUCE: The First Forty Years. In Andreas Dolzmann, Andreas Seidl, and Thomas Sturm, editors, *Algorithmic Algebra and Logic. Proceedings of the A3L*, pages 19–24. Books on Demand GmbH Norderstedt, 2005.

[90] Grete Hermann. Die Frage der endlich vielen Schritte in der Theorie der Polynomideale: Unter Benutzung nachgelassener Sätze von K. Hentzelt. *Mathematische Annalen*, 95(1):736–788, 1926.

[91] Grete Hermann. The question of finitely many steps in polynomial ideal theory. *SIGSAM Bull.*, 32(3):8–30, September 1998.

[92] Tetsuo Ida. In memoriam: Professor Eiichi Goto. *ACM SIGSAM Bulletin*, 39(3):71–72, September 2005.

[93] David Jeffrey, editor. *ISSAC '08: Proceedings of the twenty-first International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2008. ACM.

[94] Richard D. Jenks, editor. *Eurosam 74*, volume 8/3 of *SIGSAM Bulletin*, New York, NY USA, 1974. ACM.

[95] Richard D. Jenks, editor. *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, New York, NY, USA, 1976. ACM.

[96] A. Karatsuba and Yu. Ofman. Multiplication of many-digital numbers by automatic computers. *Dokl. Akad. Nauk SSSR*, 145:293–294, 1962.

[97] Manuel Kauers, editor. *ISSAC '05: Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2005. ACM.

[98] Manuel Kauers, editor. *ISSAC '13: Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2013. ACM.

[99] Donald E. Knuth. Son of seminumerical algorithms. *ACM SIGSAM Bulletin*, 9(4):10–11, 1975.

[100] S. Y. Ku and R. J. Adler. Computing polynomial resultants: Bezout's determinant vs. Collins' reduced P.R.S. algorithm. *Commun. ACM*, 12(1):23–30, January 1969.

[101] Wolfgang Kuechlin, editor. *ISSAC '97: Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1997. ACM.

[102] Y.N. Lakshman, editor. *ISSAC '96: Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1996. ACM.

[103] Santiago Laplagne, editor. *ISSAC '25: Proceedings of the 2025 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2025. ACM.

[104] Peter J Larcombe. On Lovelace, Babbage and the origins of computer algebra. In Michael J. Wester, editor, *Computer Algebra Systems–A Practical Guide*. Wiley, 1999.

[105] C.Y. Lee et al. A language for symbolic communication. Technical Report Unpublished Technical Memorandum 62-3344-4, Bell Telephone Laboratories, 1962.

[106] A.H.M Levelt, editor. *ISSAC '95: Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1995. ACM.

[107] Steven Levy. *Hackers: Heroes of the Computer Revolution*. Anchor Press/Doubleday, 1984. Google-Books-ID: o3YfAQAAIAAJ.

[108] Anton Leykin, editor. *ISSAC '11: Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2011. ACM.

[109] John D. Lipson. Chinese remainder and interpolation algorithms. In *Proceedings of the second ACM Symposium on Symbolic and Algebraic Manipulation*, pages 372–391, 1971.

[110] Jan Łukasiewicz. *Uwagi o aksjomacie Nicod'a io „dedukcji uogólniającej"*. skł. gł. Księgarnia SA Książnica-Atlas, 1931.

[111] Malcolm MacCallum, editor. *ISSAC '94: Proceedings of the International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1994. ACM.

[112] Angelos Mantzaflaris, editor. *ISSAC '20: Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2020. ACM.

[113] John P. May, editor. *ISSAC '09: Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2009. ACM.

[114] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part I. *Commun. ACM*, 3(4):184–195, 1960.

[115] Charles K. Mesztenyi. FORMAL–a formula manipulation language. Technical Report 70-133, University of Maryland Computer Science Center, September 1970.

[116] Marc Mezzarobba, editor. *ISSAC '21: Proceedings of the 2021 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2021. ACM.

[117] S. Millman, editor. *A History of Engineering and Science in the Bell System: Communications sciences (1925-1980)*. AT & T Bell Laboratories, 1984.

[118] Teo Mora, editor. *ISSAC '02: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2002. ACM.

[119] Joel Moses. Algebraic simplification: a guide for the perplexed. In *Proceedings of the second ACM Symposium on Symbolic and Algebraic Manipulation*, pages 282–304, 1971.

[120] Joel Moses. Symbolic integration: The stormy decade. *Communications of the ACM*, 14(8):548–560, 1971.

[121] Joel Moses. Towards a general theory of special functions. *Communications of the ACM*, 15(7):550–554, 1972.

[122] Joel Moses. Macsyma: A personal history. *Journal of Symbolic Computation*, 47(2):123–130, 2012.

[123] Joel Moses and David Y. Y. Yun. The EZ GCD algorithm. In *ACM'73: Proceedings of the ACM annual conference*, pages 159–166. ACM, 1973.

[124] Bernard Mourrain, editor. *ISSAC '01: Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2001. ACM.

[125] Katsusuke Nabeshima, editor. *ISSAC '14: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2014. ACM.

[126] Allen Newell. Information Processing Language V manual. Technical Report P-1513, RAND Corporation, 1958.

[127] Allen Newell and Herbert A. Simon. The Logic Theory Machine: A complex information processing system. Technical Report P-868, RAND Corporation, July 1956.

[128] Edward W. Ng, editor. *Eurosam '79" An International Symposium on Symbolic and Algebraic Manipulation*, number 72 in LNCS, Berlin, Heidelberg, 1979. Springer Verlag.

[129] Vanni Noferini and Alex Townsend. Numerical instability of resultant methods for multidimensional rootfinding. *SIAM Journal on Numerical Analysis*, 54(2):719–743, 2016.

[130] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, 2003.

[131] Joris van der Hoeven and Mark van Hoeij, editors. *ISSAC '12: Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2012. ACM.

[132] J. A. van Hulzen, editor. *EUROCAL '83: European Conference on Computer Algebra*, number 162 in LNCS, Berlin, Heidelberg, 1983. Springer Verlag.

[133] Arthur C Norman. Computing with formal power series. *ACM Transactions on Mathematical Software (TOMS)*, 1(4):346–356, 1975.

[134] S.R. Petrick, editor. *SYMSAC '71: Proceedings of the second ACM Symposium on Symbolic and Algebraic Manipulation*, New York, NY, USA, 1971. ACM.

[135] Allen Reiter. *Automatic generation of Taylor coefficients (TAYLOR)*. Mathematics Research Center, United States Army, the University of Wisconsin, 1965.

[136] Daniel Richardson. Some undecidable problems involving elementary functions of a real variable. *The Journal of Symbolic Logic*, 33(4):514–520, 1969.

[137] Robert H. Risch. The problem of integration in finite terms. *Transactions of the American Mathematical Society*, 139:167–189, 1969.

[138] Robert H. Risch. The solution of the problem of integration in finite terms. *Bulletin of the American Mathematical Society*, 76:605–608, 1970.

[139] Daniel Robertz, editor. *ISSAC '15: Proceedings of the 2015 ACM International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2015. ACM.

[140] Markus Rosenkranz, editor. *ISSAC '16: Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2016. ACM.

[141] Maxwell Rosenlicht. Liouville's theorem on functions with elementary integrals. *Pacific Journal of Mathematics*, 24(1):153–161, 1968.

[142] Maxwell Rosenlicht. Integration in finite terms. *The American Mathematical Monthly*, 79(9):963–972, 1972.

[143] Jean E. Sammet. Survey of formula manipulation. *Commun. ACM*, 9(8):555–569, August 1966.

[144] Jean E. Sammet. Formula manipulation by computer. In *Advances in Computers*, volume 8, pages 47–102. Elsevier, 1967.

[145] Jean E. Sammet. *Programming Languages: History and Fundamentals*. Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

[146] Jean E. Sammet. Programming languages: history and future. *Commun. ACM*, 15(7):601–610, July 1972.

[147] Jean E. Sammet. personal communication, 1990s.

[148] Jean E. Sammet (presumed), editor. *SYMSAC '66: Proceedings of the first ACM Symposium on Symbolic and Algebraic Manipulation*, New York, NY, USA, 1966. ACM.

[149] Tateaki Sasaki. An arbitrary precision real arithmetic package in REDUCE. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 358–368. Springer, 1979.

[150] Arnold Schönhage and Volker Strassen. Fast multiplication of large numbers. *Computing*, 7:281–292, 1971.

[151] J.R. Sendra, editor. *ISSAC '03: Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2003. ACM.

[152] James R. Slagle. *A heuristic program that solves symbolic integration problems in freshman calculus: symbolic automatic integrator (SAINT)*. PhD thesis, Massachusetts Institute of Technology, 1961.

[153] Cynthia Solomon, Brian Harvey, Ken Kahn, Henry Lieberman, Mark L. Miller, Margaret Minsky, Artemis Papert, and Brian Silverman. History of LOGO. *Proceedings of the ACM on Programming Languages*, 4(HOPL):1–66, 2020.

[154] Alma Steingart. *Axiomatics: Mathematical Thought and High Modernism*. University of Chicago Press, 2023.

[155] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4), 1969.

[156] Alfred Tarski. A decision method for elementary algebra and geometry. In *Quantifier elimination and cylindrical algebraic decomposition*, pages 24–84. Springer, 1948.

[157] Andrei L. Toom. The complexity of a scheme of functional elements simulating the multiplication of integers. *Doklady Akad. Nauk SSSR*, 150(3):496–498, 1963.

[158] Carlo Traverso, editor. *ISSAC '00: Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2000. ACM.

[159] M. Pilar Vélez, Tomás Recio, and Carlos Ueno. Niagara Falls and the origins of computer algebra. *Maple Transactions*, 2(1), September 2022.

[160] Martinus J. Veltman. A CDC 6600 program for symbolic evaluation of algebraic expressions. Technical report, CERN, 1963.

[161] Paul S. Wang, editor. *SYMSAC '81: Proceedings of the fourth ACM symposium on Symbolic and algebraic computation*, New York, NY, USA, 1981. ACM.

[162] Paul S. Wang, editor. *ISSAC '92: Papers from the International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1992. ACM.

[163] Paul S. Wang and Linda Preiss Rothschild. Factoring multivariate polynomials over the integers. *SIGSAM Bull.*, 28:21–29, December 1973.

[164] Paul S. Wang and Linda Preiss Rothschild. Factoring multivariate polynomials over the integers. *Mathematics of Computation*, 29(131):935–950, 1975.

[165] Henry S. Warren. *Hacker's Delight*. Addison-Wesley Professional, 2002. Google-Books-ID: h3zvwQEACAAJ.

[166] Shunro Watanabe and Morio Nagata, editors. *ISSAC '90: Proceedings of the International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1990. ACM.

[167] Stephen M. Watt, editor. *ISSAC '91: Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1991. ACM.

[168] Stephen M. Watt. On the future of computer algebra systems at the threshold of 2010. In *Proceedings of the Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium of Computer Mathematics and Mathematical Aspects of Computer and Information Sciences*, volume 22 of *COE Lecture Notes*. Kyushu University, 2009.

[169] Stephen M. Watt, editor. *ISSAC '10: Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 2010. ACM.

[170] David Y. Y. Yun. *The Hensel lemma in algebraic manipulation*. PhD thesis, MIT, 1974.

[171] Hans Zassenhaus. On Hensel factorization I. *J. Number Theory*, 1(1):291–311, 1969.