Origins of the APT Language for Automatically Programmed Tools

Douglas T. Ross
SofTech, Inc.
Waltham, MA

1. INTRODUCTION

(From THE NEW YORKER)

> Cambridge, Mass., Feb. 25 --
>
> The Air Force announced today that it has a machine that can receive instructions in English, figure out how to make whatever is wanted, and teach other machines how to make it.
>
> An Air Force general said it will enable the United States to "build a war machine that nobody would want to tackle."
>
> Today it made an ashtray.
>
> -- San Francisco Chronicle

... In a sulk, probably.

This quote from the New Yorker Magazine of March 28, 1959 resulted from a newspaper article, from a wire service story, covering a press conference held at the Massachusetts Institute of Technology, February 25, 1959. (Souvenir aluminum ashtrays, milled in three dimensions by the first APT System, were indeed included in the press kit!) The press conference, jointly sponsored by the Aircraft Industries Association (AIA, now the Aerospace Industries Association), the Massachusetts Institute of Technology (MIT), and the Air Materiel Command (AMC) of the United States Air Force was attended by over 60 members of the technical and popular press and resulted in extensive coverage throughout the world for the next few months. The widespread attention which this event received was not misplaced, for it represented the true inauguration of the era of computer-aided manufacturing which still is in the process of vigorous evolution today, some two decades later.

The February 1959 press conference [MIT 1959a] formally marked the completion of the opening phase of APT System development and roughly the midpoint of my own active involvement in the creation and determination of the basic structure of APT. APT stands for "Automatically Programmed Tools", and the APT language, in more elaborate form, is now an international standard for the programming of numerically controlled machine tools.

Although I have been almost overwhelmed by the amount of work involved, I am both honored and pleased with this opportunity to present a personal summary of those hectic and historic times. When first invited to prepare this paper, I envisioned a simple set of reminiscences which would be fun and easy to do. But when confronted with the almost 70 shelf feet of archival working papers, reports, and records which I have retained over the years, (two large file cabinets worth of which concern APT), I realized I was faced with a major undertaking. I have tried to minimize my personal opinion as much as possible by referring back to the original source documents which exist in such profusion. I hope I have done this in a way that is both entertaining and instructive.

I hope that I will not be accused of immodesty for writing this paper in the first person, but quite frankly I am not an impartial historical scholar and know of no other way than to provide this personalized account. I hope to share with you some of the excitement of discovery that comes from the detective work needed to place specific undated bits and pieces into what must have been the actual sequence -- long since forgotton -- as supported by evidence in the source material itself. I have been surprised many times by the evidence that discoveries and observations were made years earlier than I would have guessed or remembered, and I think these provide a useful and enlightening commentary on the historic roots of many concepts of computer science and language design which today are recognized as of central importance. I will try to point these out when they occur in the story.

1.1 An APT Theme for This Paper

There are many stories that could be written about APT. By its very nature and timing, APT simply had to be an exciting, dynamic, driving, multi-threaded, and multi-faceted brew of historic firsts -- and it was. This means that any story of APT must be complex, for the historic truth of what happened from 1956 to 1960 was indeed complex. Conceptual, technical, technological, organizational, and political problems, each of which could form the basis for a historical paper, all were intertwined and all

were influential on each other. Therefore, in order to treat the subject of APT language development as a separable component in a way that will provide some useful perspective from the present day, looking back in time, I have tried to select a theme which will allow us to see some pattern relevant to this one aspect, without doing violence to the validity of the historical record. I will try to state that theme briefly here and to refer back to it with fair regularity as my story unfolds, but I am sure that my effort will only be partially successful. So many non-language topics must be treated in order to characterize the subjects and issues that are of thematic importance that I am sure that the theme itself will occasionally be swamped, shortly to resurface like an intrepid white-water canoeist.

My theme itself is a discovery, not an invention, and is a direct result of my many weeks of struggle through the mass of material available to me. Although 20 years had completely blanked it from my memory, I found that at the very beginning of the APT effort I had an extremely clear and "modern" view of both the nature and substance that the language should have well before any real substantive progress had been made. This surprised me. I was not, however, surprised at the many high and low points, that showed clearly in my reconstruction of the activities of building the initial APT system itself. Those parts I did recall, in flavor at least (although again many of the actual timings surprised me and intrigued me in my reconstruction efforts). But when I finally got to examining the first complete APT language presented in the Phase I APT Part Programmer's Manual, of early 1959 [Ross 1959a], I was struck by the observation that although the funny punctuation made the language somewhat stilted, nonetheless an amazing number of the original high-flown objectives for the language had in fact been achieved. Out of the white-water hurly-burly of those two years of hectic project activity had emerged a reasonably user-oriented, English-like language. Again, the canoeist is a valid analogy.

My attempt to understand this phenomenon -- how with no valid precedents to go on and with the most chaotic possible working environment (with essentially no time for research, but only breakneck development) this desired end result came about -- supplied my theme. When I asked myself why did the APT language turn out to be "English like" and well-suited to the needs of its part-programmer users in spite of the fact that there was no theoretical basis for language design or implementation or management of large, dispersed, team programming efforts, I could see only one pattern which emerged from the historic record which I had reconstructed. That pattern, or theme, seems to have four primary premises. These premises, combined with the factual observation that our efforts were indeed successful, yield a conclusion which seems to explain the reason for that success. The four premises are:

1. The entire field of automatic programming for numerical control was brand new. Therefore, with respect to language design, the semantics of the language had to come first and the syntax of the language had to derive from the thinking or viewpoint engendered by technical ability to have a "systematized solution" to the general problem area.

2. For many pressing reasons, the project and the work to be done had to be broken into many pieces to be worked on by many people at different locations. This implied that even though there were no guidelines to go by, some form of viable modularity had to be imposed on every aspect of the solution. The system had to have a structure mirroring the structure of the systematized solution.

3. Lacking today's compiler-building technology, both the syntactic (translation) and semantic (execution) processes had to be cohesive modular units within the system structure.

4. In order to satisfy the requirements for the system and language as a whole, both the syntactic and semantic aspects of both the language and the system had to be open-ended, so that both the subject matter and the linguistic treatment of it could be extended as the underlying manufacturing technology evolved. In particular, the system had to be independent of geometric surface types, and had to be able to support any combination of machine tool and control system.

These four premises, which I will cite when they occur in the record and which were religiously observed from the beginning, combined with the fact that we were indeed able to get the job done, seemed to (and this is the theme) supply the equivalent of a theoretical framework matching almost feature for feature the current high-technology, theoretically-founded approach to building advanced man-machine computer application systems. In other words, even though we stumbled frequently along the way, and at times created rather awkward and juvenile versions of what we now accept as required, nonetheless we did manage to get all of the essential pieces done in essentially the right way, in essentially the right places so that even today some twenty years later, the structure of both the language and the system are essentially the same and have been found to have stood the test of time well. In effect we were forced by the nature of our problem and the necessity of our approach to it to anticipate much of the well-structured methodology that from today's vantage point seems essential and almost inevitable. As we will see,

however, the first time through, that inevitability was not quite so obvious!

1.2 An Overview

Before commencing on the story itself, I will present a brief overview of the rough chronology of times, places, people, and events to provide a bit of orientation. Much as different periods of development can be observed, after the fact, in the evolution of style in painters and artists, the beginnings of APT also break naturally into a number of overlapping but nonetheless distinctive periods during which the focus of attention highlighted first one and then another and then another aspect of the overall subject.

The first major period covers roughly September 1956 to January 1957, during which I and my green staff at MIT came up with the first overall formulation of the APT System concept and documented it in the first Interim Report of the project [Ross and Pople 1956]. We laid out a system structure distinguished by three styles of three-dimensional programming (by points, by space curves, and by regions) and a system structuring technique based upon the simulation of a specialized APT computer on a general purpose computer. My coverage of this first period here is preceded by historical background with respect to numerical control and the MIT Milling Machine Project, as well as of my Computer Applications Group. Since some of the system and language design objectives which formed the starting point for my theme come from background working papers generated during the preparation of the Interim Report, a couple of side excursions set the tone of my thinking as the project actually got underway.

To me a very interesting pair of discoveries which I never would have guessed if I had not found the evidence in my papers, is the fact that my concepts of plex programming via "reversed index registers" (and the modern equivalents of beads, records, controlled storage classes, and abstract data types) as well as the essential difference between "input" and "control" of my 1970's formulation of plex ideas in the form of Structured Analysis both were recognized as important and documented by me way back then in the fall of 1956. These are among the events that form the source for that first Interim Report formulation of APT.

The second period concerns our initial activities in conjunction with the Aircraft Industries Association, -- a trade association of all of the major companies involved in any way with aircraft manufacturing. We first made contact with them early in 1957 and the second period stems from their interest in having MIT put on a Special Course covering all aspects of numerical control. Their interest stemmed from the fact that a major Air Force "buy" of $30 million (yes! 1957 dollars!) worth of numerically controlled machine tools was scheduled to start, with installations in many of their plants beginning in the summer of 1957. The companies had to figure out how to put them into productive use. Although there was as yet no APT language, I present here my lecture about language design for numerical control from that course, for it also forms a cornerstone in my thinking about APT language before it actually existed.

Our APT research progress showed the greatest promise for meeting the needs of the aircraft industry in time to be of use, so that the third period covers the AIA decision to form the APT Joint Effort. This began with a kick-off meeting at MIT the week of May 20, 1957, which launched the world's first major cooperative programming venture, combining government, university, and industry, with the Air Force sponsoring MIT leadership of a 14-company team effort.

Still there was no APT language. The fourth period which I consider took place primarily on two days of a weekend (May 25, 26, 1957), followed by spot efforts over the next two weeks as I generated the first memorandum defining the initial APT language [Ross 1957c], intended merely as a starting point and designed primarily to permit the simplest possible translation approach. The memo is brief and historically interesting so I present it here in its entirety.

The fifth period concerns most the structuring of the APT System and the stages of evolution of how to treat the input translation, instruction preprocessing, and definition preprocessing portions of that system which in modern terms constitute the lexical analysis and parsing of APT language expressions. Although I don't go into detail here, I try to indicate a bit of the struggle that led to modular program structuring which from there on served the same role as the modern high-technology approach to compiler construction. I also cover progress in the computational aspects of the geometric semantics of the language which led to the first paring down of expections to a Field Trial system appropriate for testing.

The sixth period concerns the construction of the Field Trial APT language, with side excursions into the preparation of the first two major papers describing APT, one presented at the Third Annual Contour Machining Conference [Ross 1957d], and the other at an ACM session of the American Association for the Advancement of Science (AAAS)[Ross 1957f]. Extracts from these papers once again provide a concise expression of the kind of thinking that lay behind the day-to-day work.

The Field Trial distribution marked the end of MIT's official coordination of the APT Joint Effort, which from then on passed into the hands of a succession of leaders from industry. The transition was not easy, however, and difficulties continued to plague the project so that the seventh period centers around a "Post-Coordinator's Report" in the fall of 1958 [Ross 1958f] which draws together many of the syntactic and semantic problem areas and supplies solutions which completed the specifications for the Phase I APT system -- the first really complete APT system and language, which is the ending point for this historic analysis.

The eighth period covers the actual Part Programmer's Manual of the Phase I APT

language, citing how its features evolved from the initial and Field Trial formulations, through the evolutionary stages cited in the papers, into the first complete formulation. I also point out how the concommitent evolution of the syntactic and semantic processing, as covered in the Project Report and Coordinator Report descriptions, made possible the various language features, and cite how they match with many of the original high-blown objectives of the first Interim Report and the Special Numerical Control Course of the spring of 1957.

The ninth period covers the public presentation of these almost-working results in the February 25, 1959 Press Conference at MIT, and I also briefly outline some of the syntactic and semantic language features which purposely were left out of Phase I but which later came into subsequent phases of APT evolution, in several cases anticipating powerful language features (such as macros and phrase substitution) found in today's high level languages.

APT has undergone continuous evolution and extension from the beginning up to the present day and therefore I make no effort to extrapolate or to provide a tutorial summary of modern-day numerical control programming, computer graphics, and computer-aided design -- all of which are part of the same evolutionary development. But I hope that my efforts to present an organized analysis of these early developments will be found to support the theme which I discovered threaded through the historic record -- that because we were forced to be modular and open-ended, in a very difficult and brand new application area, the fact that we succeeded ensured that the resulting system and language were "modern" by today's standards, even though done some twenty years ago.

1.3 Generality in Specialty

The fact that APT was the first and most widely used of the special purpose application-oriented languages counts for something. But more importantly, it seems that APT is particularly apt as a subject for study in the history of programming languages. Being a language for a specialized area -- automatic programming of numerically controlled machine tools -- might appear to make it inappropriate, because it would appear to lack generality. But it turns out that, because the application area was brand new and never before had been attacked in any way at all, the study of the origins of the APT language necessarily involves much greater attention to semantics than is the case with respect to more general-purpose languages which obtained most of their background ready-made from the fields of mathematics and logic. There is no way to separate the origins of the APT language from the origins of numerical control itself, and of the community of users of numerical control. Moreso than for any other historic language, APT requires a complete consideration of every aspect of relevance to language design.

This perhaps surprising generality of interest in a specialized language is further bolstered by the fact that, because it involved so many firsts, including the first introduction of the use of computers in large segments of the manufacturing industry, we were forced at the time to be very explicit about our thoughts, objectives, and motivations in designing the APT language and APT system, in terms addressed directly to the intended users. Thus, among my many source documents are specific writings concerned with the very subject of the motivations behind the design of the APT language. It is not necessary to speculate at all about what was in my mind twenty years ago, for we can find it expressed directly on paper, written at that time. To me, this has been one of the most interesting discovery aspects, for it shows that we were very much aware of what we were doing and that we expressed ourselves in surprisingly "modern" terms. I will present verbatim extracts at appropriate points in my story and let the modern reader be the judge.

1.4 Source Material

As to my source material itself, it is held together and (meta-) commented on in an on-going, continuing fashion by an extensive series of "daily resumes" [Ross 1956b-1963], which were a form of professional diary which I kept with surprisingly complete regularity from

> October 30, 1956 through May 20, 1959,
> December 15, 1959 through March 10, 1960,
> December 4, 1961 through October 2, 1962,
> February 1, 1963 through November 29, 1963,

finishing with a few bits and pieces in connection with the IFIP Congress in 1965. Every day or so, and often periodically during the day, I would merely grab my dictating machine and say who I had met with about what, or what I had read that had interested me, what trips I had taken, etc. and, in general, what I was thinking about. I started this practice initially shortly after the beginning of the APT Project at MIT when it became apparent to me that I could not keep all my activities straight without some help. I decided that the time of my secretary in transcribing these daily resume notes would be well invested in allowing me to know what was going on. This blow-by-blow description -- all 833 pages of it -- is, of course, invaluable reference material for historic reconstruction.

In addition to the resumes, my files contain the expected correspondence and reports, extensive technical memoranda series, trip reports, notes, and meeting handouts, as well as many clumps of actual original working papers and working drafts which led to published material, sample computer runs, and the like. At the time, the MIT Servomechanisms Laboratory supplied us with pads of unlined yellow paper (yellow since it was non-reproducible by the Ozalid process and much of our work was classified). In my group we attempted to retain working papers for continual back-checking, and we said "Find it fast in the Yellow Pages". Only a few of them have survived, but those that have are interesting indeed.

To provide concise reference to these materials in my files (which ultimately will reside in the MIT Archives) I have used the following condensed notation: [R56123] means "resumes

1956 December 3"; [C570123] means "correspondence 1957 January 23"; [N57223, p4] means "page 4 of working paper notes 1957 February 23", etc., and [p3] means "page 3 of the most recently cited reference", etc.

2. BACKGROUND

Because of the importance of the pragmatic and semantic components of APT language design, it is appropriate to begin with a brief overview history of the events before, during, and after the time period (from the fall of 1956 through the spring of 1959) actually covered by the story of this paper. I will interweave a bit of my own history and that of my Computer Applications Group at MIT with that of numerical control itself in order to lay the proper foundation for the personalized account which follows.

I came to MIT in the fall of 1951 as a Teaching Assistant in the Mathematics Department, following my completion of an Honors Degree in Mathematics at Oberlin College in June. I started in the Servomechanisms Laboratory (now the Electronic Systems Laboratory) with a summer job in June 1952 and stayed with the lab until departing to form SofTech in July 1969. As is evidenced most accessibly by the excellent article [Pease 1952] in the September 1952 issue of Scientific American, 1952 also was the year of completion of the first phase of the introduction of numerical control to the world, with the operation of the MIT numerically controlled milling machine at Servo Lab, [MIT 1952].

As is described more completely elsewhere [Ward 1968], the development of numerical control, proper, began with a proposal by John T. Parsons of the Parsons Corporation, Traverse City, Michigan to the United States Air Force that punched tape and servomechanism control be applied to a milling machine to produce automatically the templates required in the production of helicopter rotor blades. Parsons received a contract from the Air Force in July 1949, and subcontracted the control work to the Servomechanisms Laboratory in 1949. In February 1951 the Air Force contract was switched directly to Servo Lab which was to receive continuing sponsorship for numerical control hardware, software, and adaptive control, followed by computer-aided design, computer graphics hardware and software, and software engineering and software technology, for almost 20 years.

My entry to Servo Lab was in 1952 in the field of airborne fire control system evaluation and power density spectra analyses. (I later completed a Masters Thesis on Improved Computational Techniques for Fourier Transformation [1954] as well as all course requirements for the Ph. D. in pure mathematics by taking courses throughout this period as a part-time student. I never completed the doctorate program, since qualifying examinations would have covered material in pure mathematics that I had not studied for years and there was, of course, no computer science doctorate program at that time.) The MIT Digital Computer Laboratory with its Whirlwind I computer [Everett 1951] had recently spun off from Servo Lab and I taught myself to program it in the summer of 1952, completing my fall teaching schedule while also taking a full course load and being a full-time staff member at Servo Lab.

From the fall of 1952 and for the next seven years I was heavily involved in the application of a new form of air mass ballistic tables to the evaluation of airborne fire-control systems, especially for the B-58 bomber. A series of projects eventually involved the installation of manual intervention and Charactron display tube equipment on the ERA 1103 Computer (precursor to the Univac 1100 series of computers) at Eglin Air Force Base in Florida. This work, which preceded and heavily overlapped with the early APT developments, had a very strong influence on my approach to the numerical control programming and calculation problem. The problems were of similar complexity and involved the same style of "systematized solution" using three-dimensional vector calculations, intermixing a heavy use of logical decisions with simple approximating calculations in a complex network modeling the reality of the situation [Ross and McAvinn 1958]. This also was the source of the foundations of my plex philosophy and approach to general problem modeling and solution.

In the four years from the summer of 1952 to the summer of 1956, I was involved in a broad range of small and large projects. I and my small group of programmers had developed a Mistake Diagnosis Routine for interruptive checking of arbitrary programs, and various program debugging and preparation aids. We commissioned programmers of the Digital Computer Laboratory to prepare a symbolic assembly system for preparing ERA 1103 computer programs on Whirlwind, as well as a "director tape" program for Whirlwind, which probably was the first JCL-driven operating system. We were the first to use the then-classified manual intervention and multiple scope equipment (developed for the Air Defense System) for general interactive man-machine problem solving. I wrote the first two-dimensional freehand, graphic input program (1954) starting computer graphics, and we installed the first interactive on-line typewriter keyboard (1955). All of these features including automatic logging and playback of manual actions were included in the Servo Lab Utility Routine Program (SLURP) System [Ross 1958b], including a system called "Group Control" [MIT 1959b, Vol. VII] for automatic paging between core memory and drum storage to allow large programs to operate in limited memory.

Many of these activities were simultaneously underway in the summer of 1956, including a multi-organization "Pre-B-58 Project" [MIT 1958b] for the evaluation of the tail turret of the B-58 airplane, with programmers under my direction in St. Louis, Eglin, and MIT shuttling back and forth developing both hardware and software for Whirlwind and the ERA 1103. A number of the important concepts behind this work were summarized in my first professional paper on

Gestalt programming, presented that spring at the 1956 Western Joint Computer Conference in California [Ross 1956a]. I set aside an example of push-button control of (pre-Sputnik) satellite launching, and instead illustrated that paper with an example of control of an automatic factory -- appropriately enough for APT.

2.1 The MIT Milling Machine Project

While these many activities were forming the background of myself and my programming group, the MIT Numerical Control Project was also progressing full steam on several other parallel paths from 1952 to 1956. Through numerous contacts with machine tool builders and aircraft companies, large numbers of test parts were produced on the milling machine and were studied for economic impact. Many parts were programmed entirely by hand, but throughout this period John Runyon developed a comprehensive subroutine library on Whirlwind [Runyon 1953] for performing many of the necessary calculations and automatically producing the coded instructions for the punched paper machine tool control tape. Many parts were produced by a combination of manual and subroutine means. Improvements also were made to the hardware of the control system and machine tool director during this period, along with documented economic studies and much educational documentation and promotional material.

In 1953 the primary sponsorship support for the engineering-based milling machine project switched from the Air Force to the Giddings and Lewis Machine Tool Co. which sponsored the development of a new production tool director on a commercial basis. During this period, the Air Force-sponsored economic studies were completed [Gregory and Atwater 1956] and the Air Force-sponsored engineering project commissioned the development, by Arnold Siegel of the Digital Computer Laboratory, of the first automatic programming language and system for higher-level language preparation of machine tool control tapes. Arnie did a wonderfully elegant prototype demonstration system for two-dimensional parts consisting of straight lines and circles called for respectively by symbolic names beginning with S or C and with mnemonic punctuation for indicating near and far intersection points and controlling the tool motion on the right or left side of the specified curve, see Figure 4 (later). The resulting MIT report and Control Engineering article [Siegel 1956a,b] stimulated a great deal of interest.

Based upon this success and that of the Runyon subroutine library, the engineering project submitted a proposal to the Air Force that numerical control work be continued at MIT with a concentration on automatic programming rather than further engineering. Then in 1955 the entire engineering project staff (except for Alfred Susskind, who remained at MIT for teaching and research) left the Laboratory to form Concord Controls, Inc. with Giddings and Lewis backing, to manufacture the Numericord director system on a commercial basis [MIT 1956].

2.2 The Computer Applications Group

The Air Force accepted the Servo Lab proposal. Beginning in June of 1956 the laboratory was under contract to inaugurate a new program in numerical control, this time emphasizing automatic programming for three-dimensional parts to be produced by 3- and 5-axis machine tool director systems. Naturally enough, this work was assigned to my group. In recognition of the many projects which we already were serving in addition to the milling machine project, we were commissioned as the Computer Applications Group, at that time. Since I was understaffed for the work already underway, an entire new recruiting effort was immediately instituted. John Ward, then Executive Officer of the Laboratory was acting Project Engineer, but technical responsibility fell to me.

Late in the summer we hired our first programmer trainee, Jerome Wenker, but actual work on the project was forced to wait until the arrival in September of three new Research Assistants with engineering backgrounds. They were Samuel M. Matsa, Harry E. Pople, Jr., and George Zames. Thus the revitalized milling machine project got underway in September 1956, three months late, with everybody including myself completely inexperienced in numerical control and, except for me, also inexperienced in either programming or language design. Furthermore, the contract called for quarterly Interim Engineering Reports, the first of which was to report on our progress from the start of the contract through September 30! Hardly an auspicious beginning for so ambitious an undertaking, especially since the Pre-B-58 Project was beginning to encounter the first of a long series of great technical difficulties, so that that project forms the bulk of my daily resumes for many months to come. We were all bright, eager, and hardworking, however, and set to work immediately with enlightened if somewhat naive vigor.

3. PERIOD 1: INITIAL APT FORMULATION (September 1956 through January 1957)

3.1 Changes from the Siegel System

My date stamp on my own copy of Arnie Siegel's MIT report reads "September 28, 1956". Arnie was a good friend and I'm sure I met several times with him to go over his programs and gather his thoughts on how we ought to proceed. His system was based upon a simple character-driven Input Translation Program like those used elsewhere in the symbolic assemblers that had been written at the Computer Lab. The translator then selected from a library of closed subroutines which analytically computed the required intersections of straight lines and circles and employed the same calculations as Runyon's cutter center offset routines and punched-tape code-generation (recorded on magnetic tape for offline punching). His system also produced meaningful error diagnostics and was elegant and easy to use.

I saw immediately that although the input and output portions of the Siegel system could be extended, the language and calculating methods would have to be completely different, for in order to handle arbitrary 3-dimensional shapes the calculation of curves and endpoints by closed-form solutions would be impossible. Furthermore, the "automatic programming language" (it was still several months before the term "part programming language" was introduced in APT, probably by Boeing) would have to be considerably more elaborate than that used in Arnie's little prototype. I was familiar with parametric methods, having completed a course in differential geometry as part of my mathematics training, but planes, spheres, cones, cylinders, and quadric surfaces were the primary building-block shapes which had occurred in the parts that had been programmed thus far. Airfoil sections and smooth faired surfaces described by arbitrary meshes of points in space were also handled by special Whirlwind routines [Bromfield 1956]. Therefore, the more natural approach seemed to be to describe the three-dimensional motion of a cutter through space by means of a space curve resulting from the intersection between two three-dimensional surfaces.

From the extensive use of vector methods in the fire control system evaluation work, and from the natural match between a rectilinear coordinate system for calculation and the rectilinear control directions of the machine tool, it was quite natural to base all considerations on a three-dimensional vector approach. As I recall, Arnie was mildly disappointed to find that such major surgery would be required on the framework of his prototype system, but he agreed that, especially since the closed form mathematical solutions were impossible, the changes were necessary. Unfortunately, except for these few startup contacts, Arnie had no involvement with our project except for helping with the course the following spring (Section 4), and the Kick-Off Meeting (Section 5.3).

In any case, the earliest written record about APT that I find in my files is a letter [C561010] from me to Joseph Albert, then an engineer in the Manufacturing Research Department of Lockheed Aircraft Corporation, Marietta, Georgia, in reply to his October 4 letter to the Lab Director, Professor J. F. Reintjes inquiring about our use of Whirlwind for preparing tapes for our milling machine. After referring him to the relevant reports (including Arnie's, but suggesting that Runyon's report "would be of the most immediate interest to you") I then describe our new method "which is now in its infancy [and] is restricted to the case of planes and spheres, although a direct extension to include cones, cylinders, surfaces of revolution, and quadratic [sic, not the correct "quadric"] surfaces is planned. A report describing this work is scheduled for publication." -- so we were well underway.

3.2 Project Start-up

The yellow page draft Interim Report material appears to start with some notes spoken into the dictaphone [N569?] in the process of a kickoff meeting of some sort, probably in late September 1956. The notes mention starting off with only planes and spheres, using them to both define curves and program with regions, and also include the ideas of directed distance and normal vectors "to get somebody's surface in" [p1].

> "We will put in only as much vocabulary into the language as we presently need to motivate the study and make it workable for ourselves. The making of this thing into a really useful system, since we have limited manpower, we would like to have assistance from other people in establishing a vocabulary for the language, in other words, which particular surfaces are to be used or how surfaces are to be specified in particular... Now here is a problem - How do you work out a way of talking about a part in terms of regions? - What are the necessary things are not self-explanatory and that's where we will be working [p2]...
>
> "First of all we want to establish that the problem is one of language. Then, what kind of language? Well, we want our language to be just between the man and his problem, independent of the particular machine tool that is going to be used but, ... we do have to compromise somewhat on this ideal language. In other words, our language will be influenced by the particular tool and by the particular computers to some extent, but our goal is to have it as nearly independent of these quantities as possible..." [p3].

Certainly not written or dictated English, but it is a record of our early thinking and of the beginning of the APT language. This establishes premise number 4 of my theme (Section 1.1).

3.3 System Structure and Semantics

Evidently starting from this initial group meeting, sometime in September or October I prepared four pages, single spaced, of typewritten outline with corrections in ink [N5610?] starting out:

> "I. Introduction
>
> "We now give breakdown of problem into major areas as reference for all future reports and work..."

The contents verify that from the beginning I had the view that the (as yet unnamed) APT system would be modeled on the structure of a computer, with an input translation program preparing pseudo instructions to be interpreted by a control element program which in turn would select the modes of beavhior of an arithmetic element program producing a sequence of cut vectors to be output in the language of a chosen machine tool and machine tool director system. The arithmetic element program (or ARELEM, as it is now known) calculates cut vectors just as an ordinary computer calculates numbers as answers. These yellow-page outline notes are of historic interest

because they document the evolution of terminology to that still in use in APT today. The section [p1] entitled "II. Progress" reads as follows:

"Have chosen to work on PD [part description] of simulated computer first because all other parts hinge on its form.

Method of Part Description: At Computer Level: by point moving in some pattern over the surface of the part, i.e., view as point cutter. Viewed as point moving in steps by sequence of straight line cut vectors, but need not imply that actual cutter will use linear interpolation. This view merely gives discrete points on the surface and space between points may be treated in many ways depending on particular director used. Point by point calculation is required for digital treatment of problem. The sequence of cut vectors are [sic] made to approximate an elementary segment of a determined curve, defined as follows:
The point cutter is to move on some determining surface but staying [p2] in the driving plane.
The intersection of the driving plane and the determining surface is the determined curve.
The elementary segment of the determined curve is determined by the intersection of the determined curve with the check surface, the end point being called an intersection point.
Thus an elementary segment is defined by a DP, a DS and two CS's., as the section of the determined curve between the two intersection points."

After a few more lines about the simulated computer, we find [p2]

"Operation Code - the operations which the simulated computer can do. For present, simplified case

DS =
DP =
CS =
GO !
Tolerance =
END "

A very interesting point is that this paper has black ink pen corrections which include changing the first two instructions to "PS =" and "DS =" and these corrections are carried through the rest of the pages. This shows that the current APT system and APT language terminology of part surface, drive surface, and check surface, see Figure 1, resulted in this fashion as an improvement over the original idea sometime on or before November 2, 1956 (for which my resume says "The introductory part of the milling machine report now seems to be in fairly good shape. Progress on the body of the report is up to the equations for part surface = sphere" [R56112], so that the term "part surface" was established by then).

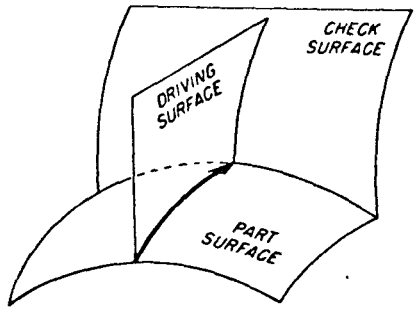


Figure 1. Space Curve Determined by Surfaces

The remainder of the outline [N5610?] contains no further language features except for a discussion of

"ways to automatically determine which direction to take when there exists two possibilities... Decided to use just K1·K2 > 0 to define 'forward' and have 'go forward!' plus 'go reverse!' modes of 'GO!'. [If the dot product is zero] then forward = right if think of standing on K1 with head in direction of N1 and look in direction of K1. This is easy to remember and can be used to treat arbitrarily complex parts." [p3, 4]

The meaning of forward, right, and left being based upon orientation with respect to the surface normal still is true of APT language, but later underwent several versions of implementation after this initial definition, as we will see. In any case, the outline pages show that although the problem was averred to be one primarily of language, it is clear that we approached the problem from the semantics of that language, not its syntax, from the beginning. This establishes premise number 1, of my theme (Section 1.1).

3.4 Generalized ARELEM

The heart of APT semantics is the Arithmetic Element program, ARELEM. We did not stay very long with the very simple plane-and-sphere case. The November 9 resume states "Jerry Wenker started thinking about a general design for part description. It involves an iterative technique for using normals to surfaces as well as programs for directed distance to surfaces to handle arbitrary situations. I talked with him about this and we developed several improvements on the technique and I pointed out that it was very similar to what we were now doing in the [Pre-B-58] evaluation program..."[R56119]. This is an important date for it marks the beginning of ARELEM developments that were not to receive really satisfactory resolution until our Second-Order MIT ARELEM was finally developed in 1961 -- well after the end of the story I tell here. But by November 19, I "wrote section on the new

elementary calculation for milling machine report" [R561119], so by this date we knew the complete structure of the intended new system.

3.5 Glimmerings of Plex

Two weeks later "most of the day was spent in milling machine conference considering primarily the use of index registers. The results are summarized in Milling Machine Conference Number 1" [R561129,30]. When I first saw that in the resume, I made the side note "I don't know whether those notes are still available. I wonder if even this early I had the idea of 'reverse index registers' or Plex programming [Ross 1961]. If so, it predates my even hearing of McCarthy's LISP system (in 1959)" [Ross 1977b]. This note was written in October 1977 before discovering that I did indeed still have the yellow-page minutes of that conference with my crew, from which the following extract is pertinent [Ross 1956c]:

> "It is desirable to be able to program by areas instead of by curves as we now do. This we feel may be accomplished by allowing an arbitrary number of surface registers, i.e., the locations for part surfaces, check surfaces, and driving surfaces. The motion over the surface of a part, whether by zig-zag or by spiral motion, will be called facing. The use of many surface registers allows a facing to have an arbitrary boundary [p1] and contain an arbitrary number of intermediate part surfaces. In order to have compact representations of complicated facing motions, we need to have indexing capabilities.
>
> "Indexing may be done on many levels and in many ways. The quantities which may be indexed are program addresses, data addresses, or data. When an instruction is tagged for indexing it is tagged whether the index is to be modified or not modified before the indexing operation, and modification always applies a check against an associated criterion. The instruction is also tagged whether the check with the criterion should cycle, terminate, or start. If cycled, then the index is reset and the instruction remains active. If terminate, the index is not reset and that instruction is ignored until the index is reset by some other means. If start, the instruction is not executed until the criterion is reached.
>
> "All indexable quantities (which are all quantities) have stored with them an increment in their appropriate units."... [p2]

The notes go on to describe the grouping of surface data together for such accessing so it does seem that what later was to become the "n-component element" and "bead" programming with pointers of AED [Ross, Rodriguez, and Feldman 1970], the "records" [Wirth and Hoare 1966] and "abstract data types" in other languages [Liskov, Wulf], and even the "controlled storage classes" of PL/1 [ANSI 1976] were recognized by us as useful in 1956. In fact, the incorporation of the logical control information in a complete cross reference network for problem solution (which is further born out by the subsequent four sides of yellow-page working papers [N56121,2] which I generated on December 1 and 2, 1956 over the weekend, leading to a "new method of language and driving surface derivation" [R56123] is still my understanding of how the "guarded commands" of Dijkstra [1975] should be thought of and used. Thus twenty years is both a long time and a short time in the history of programming.

3.6 Language Intentions

While these semantic-oriented AREL EM developments were going on, work on the Interim Report continued. In a November 7, 8 visit, our Air Force contract monitor, Bill Webster, requested that the Interim Report "should be as up to date as possible and not terminate at the September 30 date. This will entail an appreciable amount of additional writing" [R56119]. I have a yellow-page handwritten pencil draft [N561215,16] written over the weekend on December 15 and 16, 1956 addressed specifically to the "point of view of language being the most important item and the system reflects the structure of the language" [R561214+17]. The draft itself was never used, but for our purposes here, of understanding the state of mind with regard to language design at that time, a few excerpts are illuminating.

> "The objective is to instruct the machine tool to perform some specific operation, so that sentences similar to the imperative sentence form of English are required... Declarative statements are also necessary. Examples of declarative sentences used to program a numerically controlled machine tool might then be of the form:
>
> 'Sphere No. 1 has center at (1,2,3) and radius 4'
>
> 'Airfoil No. 5 is given by equation...'
>
> 'Surface No. 16 is a third order fairing of surface 4 into surface 7 with boundaries...'
>
> An imperative sentence might have the form:
>
> 'Cut the region of Sphere No. 1 bounded by planes 1, 2, and 3 by a clockwise spiral cut to a tolerance of 0.005 inch.'
>
> These sample sentences, although written here in full English text for clarity, are indeed representative of the type of language toward which the present study is aimed" [N561215,16, p2].

The draft continues to discuss the vocabulary and "syntactical rules for combining words to express and communicate ideas. In general terms, the syntactical rules determine the structure of the language, while the vocabulary

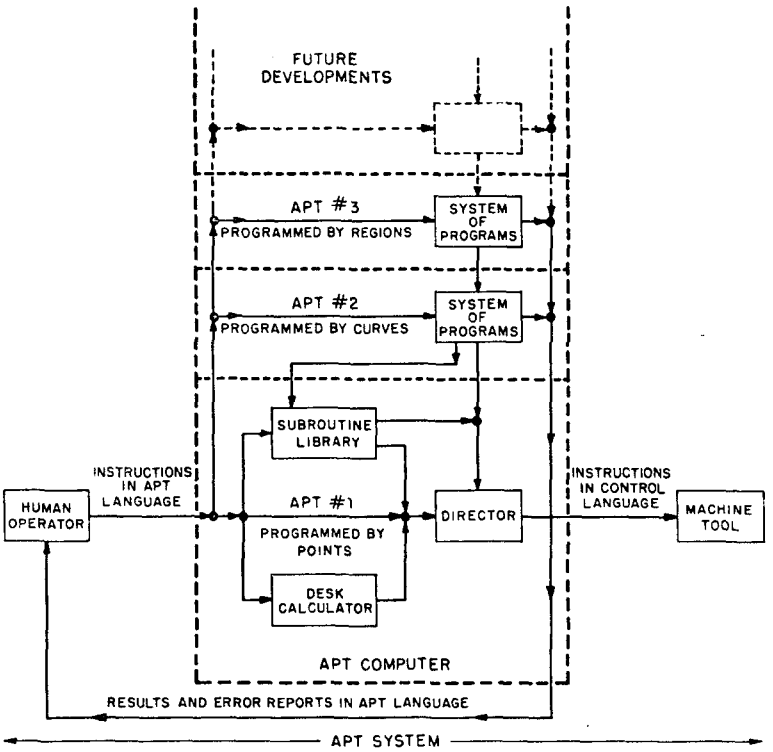


Figure 2. The APT System Concept

determines the substance of the language [p3]. ... All of the elements of syntactical structure and each word in the vocabulary of the language must be matched by an appropriate computer program. Therefore the structure and substance of the language influence directly the system of computer programs" [p4]. This establishes premise number 3 of my theme (Section 1.1).

3.7 The Name "APT" and Glimmerings of Structured Analysis

The last portion of this December 15, 16 draft shows that it was at this time that I first came up with the acronym "APT". The pages I have end with "The APT System Concept", as follows:

> "The preceding sections have introduced the basic approach to the problem of programming of machine tools, from an abstract and ideal point of view. It was mentioned above that the ideal language must be modified in practice to suit the particular machine to a computer, as well as to suit the human and his problem. A very instructive and helpful visualization of these influences may be gained by considering the practical problem from a system point of view. The general structure of an Automatic system for the Programming of machine Tools (abbreviated APT system) is shown in Fig. 1. The 'variable parameters' for APT system design are shown as vertical inputs to the principal system components -- the human and the computer " [N561215, 16 p5, 6].

Thus was the APT system christened, although in the final December 27 [R561227] rewriting of the report I say "... called APT systems (an abbreviation of Automatically Programmed Tool systems)' [Ross and Pople 1956, p4] which holds to this day.

I actually do not have the original "Figure 1", but instead have the new figure (shown here as Figure 2) which was used in the final Interim Report [p5] and elsewhere. I do, however, have a yellow typed page, with carbon [R5612?], which talks about the original Figure 1 and which I clearly wrote in the intervening ten days. It is of interest in its own right because it shows that the "variable parameters" of the original Figure 1 (which was closely related to Figure 2) were in fact the all-important "control" of my latest creation -- Structured Analysis (circa 1973) [Ross 1977a].

> "The characteristics of any particular APT system will depend upon the vertical 'inputs' shown -- the experience, design practice, and APT system capabilities which influence the human, and the various programs which control the computer. Conversely the development of a particular system with certain prescribed characteristics is centered in the study or design of these 'inputs'. Thus, the overall task can be broken into convenient sections by considering each of these 'inputs' as a separate problem. The solution of any of these problems will of course be influenced by the interdependence imposed by the overall system" [R5612?].

This establishes premise number 2 of my theme (Section 1.1), so all four premises were, as I said, observed from the beginning.

The historically interesting fact is that I clearly understood that there was a profound difference between "inputs" (without quotes) and "'inputs'" (with quotes), including their vertical placement, and that they led to hierarchic modular design. The December 14 resume says "Spent all day hashing over the Milling Machine Report trying a number of new introductory sections. A number of good ideas were brought up but were too difficult to whip into shape, such things as a hierarchy of programmable systems, etc." [R561214]. This was immediately preceding my preparation of the above-mentioned handwritten draft [N561215, 16], that weekend.

Although I have stated in my writings about Structured Analysis that it had its roots in my early plex thinking, I have attributed the four-sided box notation (with its input, control, output, and mechanism meanings for the four sides) to the "cell-modeling" notation of Shizuo Hori [Hori 1972] (who headed the APT Long Range Program follow-on to our early work for many years [Dobe 1969]). Although I am sure that neither Shiz nor I were aware of it, it might be that he was influenced by my Figure 2 and the way I talked about APT, for I did indeed talk with him during his own early developments of his ideas. But in any case it is clear why in the 1970s I found the four-sided box notation so attractive and so natural as an expressive medium for my own plex notions when I came to formalize them as Structured Analysis. I have always claimed that all of my work merely tries to solve the same philosophical problem in working form -- the problem of plex [Ross 1961 and 1975] -- which really started to form in my mind in the early nineteen fifties and was quite important in my thinking during this formative APT period.

So by the end of December 1956, the name, nature of language, method of semantic calculation and the general system structure of APT was available. In the report we even said

> "The above sequence [of APT systems programmed by points, curves, and regions] has not been terminated since there is not necessarily an end to the growth potential of the APT computer. Conceivably it can grow until all design and analysis for machined parts is included in the automatic programming system, and the APT computer is programmed simply by describing the function which a machined part is to perform" [Ross and Pople 1956, p 4].

--a clear call for what later became computer-aided design [Ross 1960; Coons and Mann 1960]. APT I, programmed by points, never existed, but at MIT we did proceed to make both APT II, programmed by space curves, and APT III, programmed by surface regions, on the Whirlwind I computer [Ward 1960], see Fig. 2.

4. PERIOD 2: SPECIAL N/C COURSE FOR THE AIA (December 1956 through April 1957)

The publication of the first Interim Report marks the end of the first, primarily background, phase of APT language development. The next phase, our first contact with the Subcommittee for Numerical Control (SNC) of the AIA, overlapped and coincided, for the Interim Report came back from the printer while Don Clements and I were in Dallas meeting with the AIA, January 21-25, 1957 [R570121-25, p3]. (On December 6, Don Clements had been appointed Project Engineer [R56126]. He had been working on other projects in the laboratory previously.) On November 7 and 8, Bill Webster, the Air Force Contract Monitor, and his boss Colonel Dick visited with us and Bill suggested that we contact Bernard Gaiennie of Northrop, Los Angeles, Chairman of the Subcommittee for Numerical Control of the Aircraft Industries Association, and also that we should get in touch with Kenneth Wood of Boeing, Seattle, to ask him about a report on Boeing preparations for numerical control [R56117, 8]. Numerous phone calls, letters, and meetings at MIT document not only our going to the Dallas meeting, but also AIA's request that MIT present a Special Course on Numerical Control [R56124].

During the Dallas AIA meeting, my resume says, "Frank Reintjes [Servo Lab Director] called Vic Benetar of Lockheed there in Dallas and informed them that we were now considering giving a one week course at the end of March during vacation. Frank feels that he should pick up the ball as far as organizing the course and that with a one week thing we should be able to handle it all right" [R570121-25, p2]. There follow some other items concerning the content of the course and how it would be funded, but these are only of interest to historians of numerical control as such. The course itself did indeed take place with fancy printed brochure, tuition of $275.00, etc., from Monday March 25 to Wednesday April 3, 1957 [MIT 1957b, R57218].

For the purpose of APT history proper, one interesting component of the course is the series of lectures presented by myself and Arnie Siegel in the middle four days of the course. We covered the computer programming process, using as examples a mouse solving a maze to emphasize logical programming, and finding the intersection of a circle and a parabola to illustrate systematized solution iterative calculation methods [also used in my 1958 AAAS paper, described later]. Programming principles were covered from computer architecture and binary number systems through actual programs coded in IBM 704 assembly language. Arnie described his automatic programming system and I presented APT, including curve and region programming.

The most important item for APT language background, however, was the lecture which I presented, on Friday, March 29, on "Design of Special Language for Machine Tool Programming" [Ross 1957a], which I reproduce here in its

entirety from the handout notes from the course [MIT 1957c]. Not only are most of the principles still valid today, but it shows very precisely the nature and trend of my thinking just prior to the first creation of APT language proper. In the oral lecture, itself, I illustrated the principles in geometric terms by means of a number of styrofoam models which I had made up in our shop from $6.86 worth (I still have the March 6th, 1957 purchase requisition) of styrofoam balls, dowels, and blocks, as used in Christmas store decoration [N5736]. These geometric shape description principles, describing objects as joins and intersections of simpler objects, have only come into practical use in the mid-1970's through the work of Braid and Voelker [Braid 1975; Voelcker and Requicha 1977].

As we will see, I was forced to make many compromises with these design principles when APT language design actually got underway. But I feel that even today the points made in this brief lecture still are valid for user-oriented application language design. In any case, the special course was a success in completing the technical background for APT language development, so now it is time to take up the story of the physical environment in which that development took place.

Friday, March 29 - Afternoon Session

Design of Special Language for Machine-Tool Programming

Douglas T. Ross

A. Need for Human Convenience

1. Previous lecture has shown that computers can be used to perform most of the repetitious tasks of programming machine tools, and that, using automatic assembly techniques, detailed computer coding is not necessary.

It still is necessary, however, for the human to consider masses of detail, in order to express his wishes in a form which the Subroutine Library can handle. The way he thinks about the problem is governed by the Subroutine Library.

For complicated parts the amount of detailed information required of the human can be very great. This situation is dangerous because humans cannot handle quantities of detailed work reliably.

2. It is necessary to find a more natural, less detailed way for the human to express himself.

B. Language Design

1. In order to have a truly convenient way for humans to program machine tools to make parts, we must first determine what are the important properties of parts in general.

The Δx, Δy, Δz and feedrate information required by a numerically controlled machine tool are certainly a complete set of properties for arbitrary parts, but, as we have seen, these properties are far from convenient for humans.

We must find properties which are much more general, more similar to the way humans think.

This is not an easy task, however, because the properties still must be very explicit so that the computer will be able to fill in all of the necessary details which are implied.

2. Once the properties have been found, then we can assign English-like words to them so that they are easy for people to remember and use.

There must also be very explicit rules for combining these words to make meaningful statements about parts.

These rules should also be similar to the rules of English grammar so that it is easy and natural for the human to express himself.

3. The effect of choosing properties, assigning words to them, and establishing rules for combining these words, is that a special-purpose language with very explicit meanings and capabilities has been designed for "talking" to the computer about parts to be machined. It is important, for human convenience, to allow for redundancy in this language, so that the human can express himself in whatever way is natural at the moment.

C. Suggestions For Language Design

1. At the present time, language design is an art. No scientific procedures are known and each problem area requires its own special treatment.

A number of helpful steps can be outlined, however, which appear to be applicable to most problem areas.

2. The kind of statements which are to be made about a problem area are generally of two kinds:

Declarative or descriptive statements ("say" - type statements) which establish definitions and basic relationships among the things which are to be discussed.

Instruction statements ("do" - type statements) which use the relationships and definitions, and tell what action is to be carried out.

The distinction between "say" - type statements and "do" - type statements is helpful in deciding what kinds of words and rules are needed in the language.

3. It is not necessary (indeed impossible) to consider all possible statements which will

be made in the language, since we can consider only the kinds of things we wish to say and do.

At this stage we need to know only a few representative things to say things about, and things to do.

Examples of "say" statements

The _____ is on the _____.

The _____ is the same size as the _____.

A _____ with a _____ will be called a _____.

Examples of "do" statements

_____ the _____ to _____.

_____ing followed by _____ing will be called _____ing.

If you have _____ed, _____ the _____.

4. It is also possible to establish a wide variety of modifiers for the language, knowing only a few representative things to be modified.

Well-chosen modifiers can be very important to the language since they allow complicated ideas to be expressed by simple statements. They allow variations of meaning without drastically changing the form of the statements.

Things which we say things about can be modified according to size, quantity, relationship to other things, etc. "Say" - things may also be modified by other "say" - things.

Example: There are _____ _____s on the _____ of the _____.

Things which we say to do can be modified according to how quickly, how carefully, and in what mode or version they are to be done, etc.

Example: _____ _____ly to the _____ and then _____ _____ _____ly.

5. With the kinds of "say" - and "do" - statements determined and modifiers established, the main features of the language are set.

The next step is to "flesh out" the skeleton of the language with the necessary thing vocabulary.

Usually there will be many more "say" - things than there are "do" - things, since problem areas are usually quite restricted in scope.

D. Results of Language Design

1. The primary purpose of a language is to provide a means for a human to express himself in a given problem area.

If the language design process has been successful, the vocabulary and rules for modifying and making statements will be easy to learn and natural to use.

2. The language also serves another very important function which is not as easily recognized. It helps the human to organize his thinking about the problem area.

Since the important features of the problem area are represented by the various parts of the language, thoughts about a particular problem are automatically directed and channeled to the essentials of the problem by attempts to make statements in the language.

3. A language will usually be in a constant state of growth and revision.

As more and more particular problems are encountered, additions to the "say" - thing vocabulary will frequently be necessary to introduce new objects, etc., into the language.

If it is found that an expression in the language occurs very frequently, the expression itself can be given a name to be added to the vocabulary.

Occasionally new modifiers will also be required for a particular problem. These new modifiers can be used with many of the old words in the vocabulary so that a whole new class of expressions are made possible.

Finally, the whole problem area will tend to expand with usage so that additions of "do" - things to the vocabulary, and even revision of the rules for making statements may be made.

E. Practical Aspects of Language Design

1. Although ideally language design should be influenced only by the problem area and the characteristics of humans, since the language is to be used to accomplish some task involving a computer and other machines as well, a number of other factors necessarily influence the design.

A universal factor throughout the design process is the economics involved. The advantage to be derived from a given aspect of the language must be balanced against the difficulties in incorporating that aspect into a complete and working system.

2. A written form of the language must be designed which is not too cryptic to be easily remembered and used by the human, but which is relatively easy for a computer program to translate.

The written language should also lend itself readily to transcription from hand-written copy to the input medium of the computer.

The language should be sufficiently readable and understandable that untrained personnel can process the statements reliably.

3. Translation programs must be written to recognize the meaning of statements in the language and initiate appropriate computer behavior.

Since the language is likely to be in a continual state of flux and improvement, it should be relatively easy to make the corresponding changes in the translation programs.

F. Computer-to-Human Languages

1. Whenever a language must be designed (as above) for humans to make statements to computers it is also necessary, to a greater or lesser extent, for the computer to make statements back to the human.

2. It is always necessary for the computer to inform the human of any errors which the human has made, and which the computer can detect.

These statements should be made in the same language the human uses, so that they can be easily understood.

3. Often it is desirable for the computer to report back to the human, its progress in solving the problem.

These statements may be made either in the form of written statements or pictures.

In the case of machine tool programming the computer might draw a picture on an output oscilloscope of the part being made, and give written statements about crucial points of intersection, etc., to greater precision than could be drawn.

This information allows the human to check that he has instructed the computer properly.

4. Most of the points made above about language design apply equally well to computer-to-human languages.

G. Computer-to-Machine Languages

1. Computer-to-Machine languages are influenced by a different set of conditions than languages which involve humans.

2. The primary condition is to match input and output media so that additional transcription and processing is not required.

Because of the large investment in equipment there are a great many practical problems involved, and several groups are now working on suitable standards.

3. It is permissible to have limited flexibility in computer-to-machine languages, since the required flexibility can be achieved by computer programming.

4. As a backlog of experience is built up it may prove best to make specially designed computers for machine-tool problems.

At the present time, however, not enough is known about the field so that programming of general purpose computers is probably better than attempting special hardware design.

5. PERIOD 3: THE APT JOINT EFFORT (January 1957 through May 1957)

5.1 Initial Meetings

As I have already stated, the original suggestion that we get in touch with the Subcommittee for Numerical Control (SNC) of the Aircraft Industries Association (AIA) came during a November 7, 1956, visit of Bill Webster, our Air Force sponsor, to MIT. I first made contact with the SNC Chairman, "Barney" Gaiennie on December 4th [R56123+4], and by the 18th had received an invitation to the January 21 and 22 meeting in Dallas Texas [R561218, 19]. In my reply of December 26, I suggested that "20 to 30 minutes toward the end of the meeting [be allotted to] a short oral presentation of our goals" [C561226].

Don Clements and I both attended the Dallas meeting, and although I seem not to have received the official AIA minutes, my resume [R570121-25] on returning states, "I gave them a brief description of what we were trying to do, and everybody seemed to be quite excited about the prospects, although I tried to make it clear that these things were probably a little bit in the future." A subgroup under Jerry Jacob of Sperry Rand was responsible for "the computer end of things, and a meeting is planned ... at the end of February at which Lockheed, Boeing, North American, and Chance Vought will be represented. I will plan to attend this meeting and we will try to make definite assignments for working areas at that time." In the discussion, 14 companies planned to use IBM 704 computers, and my single yellow page of notes from the meeting [N570121] has a box drawn around the phrase, "Package APT #2 with language!" -- evidently my first decision to speed up our work to help out the industry. There also was considerable discussion of the desire for a special course, as we have already considered.

The Computer Programmer's Meeting actually took place March 1st in Los Angeles, immediately following the Western Joint Computer Conference. The AIA minutes [AIA 1957a] state "Mr. Ross explained that these charts indicate a proposed program to handle the general case of automatic machine tool control programming. He invited companies interested in this effort to send experienced programmers to MIT and offered to provide guidance aimed at coding this program in an acceptable form for a particular general purpose computer. [I had first broached this idea to some Lockheed visitors at MIT on February 7

[R5727]]. While the offer was gratefully acknowledged, it was determined that most aircraft plants are presently utilizing their computing staff manpower to the fullest extent on current problems. Most companies indicated that they will closely follow the work done at M. I. T. and several may ultimately accept M. I. T. 's offer"[p3]. So the seed was sown, but the time was not yet ripe.

My own resume notes of March 4 reporting on the meeting lend a bit more color.

> "I described to them the work we have done here at MIT including region programming, and as discussion developed I proposed our generalized programming by curves system and passed around several copies of the diagram that Jerry [Wenker] had made. I finally was able to show them the distinction between a subroutine library approach to this problem and the system approach which we follow. And they all were quite happy about the system approach but could not get over the idea that it was more difficult than a subroutine library. I battled with them quite strenuously on this point urging them to pool their programmers and at least do a programming by curves system for 704s. Boeing is mostly completed with their subroutine library, however, so they are quite firmly entrenched in this way of doing things, and the feeling of the group seems in general to be that they were somehow afraid of the system approach in spite of my statements that I expected Jerry to have the system all programmed in a month on Whirlwind [for planes and spheres -- which still was very optimistic]. The net results [sic] was that several subgroups were formed, one of which will be a planning group to consider the overall problem of input translation - output translation and calculations. MIT is a member of this group along with 4 of the aircraft companies ... Am hoping for another go around in this group to convince them that the system approach is really no more difficult than the subroutine library approach and much cleaner" [R5734].

5.2 The Fateful Decision

There are almost eight weeks between the March 1 Los Angeles Programmer's Meeting and the April 23, 24 Los Angeles Programmer's Meeting, followed immediately by the April 25, 26 SNC Meeting. These are significant meetings, for on April 26, the SNC gave official approval to the idea of having MIT lead the construction of the first APT system based upon our research, beginning with a kickoff meeting the week of May 20, 1957 at MIT. In the eight weeks leading up to these meetings were 1) the preparation for and presentation of the ten calendar day Special Course at MIT, 2) the preparation for and execution of two debugging trips to Florida on the Pre B-58 fire control system evaluation project of five days and ten days respectively, 3) continual work with the programmers on the Editor Generator and Logging features of SLURP, 4) work on the APT II and APT III Whirlwind programs, and 5) the start of writing of the third Interim Report. Personnel problems were acute. On February 25, George Zames left the project to transfer to the Research Lab of Electronics at MIT, on April 15 Harry Pople left for six months of Air Force duty, and on February 25 we learned that Same Matsa would leave the project in June, at the end of the semester (he actually left in November) [R571127]. Therefore, during this period I had extensive meetings with both Mathematics and Electric Engineering Departments attempting to line up replacement research assistants as quickly as possible [R57225-57418].

I have no resume notes for the April Los Angeles meetings, but I do have five pages of penciled notes on lined yellow paper taken at those meetings [N57423-26], as well as the handouts. I don't seem to have a copy of the official AIA minutes of the SNC meeting for reference, but my recollection is quite clear regarding an anecdote which in any case is the most interesting aspect of the meeting, in my mind. On the afternoon of April 25 the TruTrace Corporation gave a demonstration of their new numerical control system to the subcommittee and invited us to a dinner at a local restaurant following the demonstration. By way of hospitality they arranged an open bar at the restaurant until the private dining room to which we had been assigned could be made ready. Unfortunately, some local group such as Rotary or Toastmaster was already occupying the room and it was not until late in the evening that our dinner was ready. In addition to the several hours of open bar (always with the continuing understanding that our room would shortly be available so "you better get your last drink while you can") the restaurant management included copious wine with the meal to compensate for the delay.

As I recall it, the first thing on the agenda the following morning was the vote to approve the launching of the APT Joint Effort! I have often wondered whether the result might have been different had the TruTrace open bar been of normal spigot rather than firehose proportions. We will never know for indeed the SNC vote was favorable and APT was officially launched. (Later in a May 10th letter to attendees at the Tru Trace demonstration [C57514], Mr. George Fry, President of TruTrace, states "I want to personally apologize to you for the unexpected delay encountered before dinner, and we hope that it did not cause you any severe problem in maintaining your schedules.") (!) It didn't. Our schedule problems were to come later!

The minutes of the Programmer's Meeting (which I do have) [AIA 1957b] summarize the main structure that I proposed for the APT simulated computer and then contained the following:

> "2. Information made available and views expressed:
>
> a) The various individual computer programming approaches of North American Aviation Co., Lockheed (Marietta), and Boeing, were examined and compared. The M. I. T.

programming concept was reviewed and it was the concensus of opinion that this approach is more efficient in the long view due to its capability to expand and take into consideration future more sophisticated time saving programming procedures. However, many points of similarity and/or compatibility were discovered among all the approaches.

b) Much of the work in programming already accomplished by individual plants is applicable, with modification, to a new superior common approach .

...

e) With early and effective work by each assignee, it should be possible to complete the project by October 1957.

f) M. I. T. was considered to be in the best position to coordinate the efforts of all plants in this joint programming effort. Such coordination would include any necessary instruction in the system concepts, determination of information on progress and results, assistance with technical problems insuring common philosophies and compatibilities, maintaining surveillance over individual progress, and coordinating the final results for distribution..." [p3]

An attachment to the minutes "First group of common manuscript and computer language terms defined by the Numerical Control Data Processing Group during the meeting held on 23-24 April at AIA Los Angeles" lists meanings for 10 basic words still found in the APT language. My second page of notes from the meeting shows that clockwise (CW) and counterclockwise (CCW) circle definitions of motion (as in Siegel's system) also were discussed, but were not included in the list. My notes also show that 56 man-months were estimated for the October 1957 target date for the first system, excluding project coordination by MIT and parallel efforts for the IBM 650 computer. Ultimately AFML did partially sponsor IBM to develop a pared-down version APT, called ADAPT, for the 650 [IBM 1963]).

Upon returning to MIT from the meetings, I was greeted by a letter from J. B. (Bill) Rankin of Convair, Forth Worth stating "Since we returned from the course at M. I. T., Numerical Control activity has intensified at Convair, Ft. Worth, to such an extent that I had to miss the AIA meeting in Los Angeles. The purpose of this letter is to tell you that we have generated an acute interest in your APT program. However, we do not want to be the only participant ..." [C57429], to which I replied post haste that they had been assigned to work on the "Preprocessing portion of the A. I. A. APT II system, with counsel from MIT" and that I thought the October date was "realistic" although "It will not be easy to coordinate the efforts of so many different organizations, but with the fine spirit shown by all concerned I think we should be able to succeed quite nicely" [C57429]. Programmers are perennially optimistic.

5.3 The Kickoff Meeting

My May 1, 1957 memorandum "Preparations for Joint Programming of AIA APT II System" invitation to the Kickoff Meeting described the nature of the system, mentioned that the project would operate under "a semi-democratic procedure", and laid out an agenda for the May 20 through May 24 meeting, covering both technical topics as well as details of how the project itself would be organization [Ross 1957b]. At the meeting itself, I explained that "semi-democratic" meant that we would have full democratic discussions as long as I felt progress was being made adequately and we were on track, but when I didn't think so, I would invoke my semi-democratic authority and we would do things my way -- we had a job to do. I don't think I abused this authority but I did indeed invoke it at various times even during the Kickoff Meeting itself [R57520-24, p2]. Throughout its long history, the APT Project had a strong sense of mission and a high esprit de corps. The people involved were eager, talented, and very good to work with.

My own files actually contain very little information on the Kickoff Meeting itself. I have only one page of resume summarizing the meeting and some notes titled "Working Group - Translation Meeting, Thursday AM Session May 23, 1952" which is a very rough transcript (evidently) of a Dictabelt copy of a tape recording made during the session [N57523]. My resume states

"We had a good turnout for the joint programming effort for the A. I. A. APT II. One difficulty, however, was I was plagued and beseiged by intestinal and digestive difficulties for the whole week but Don [Clements] and Arnie Siegel and Dale Smith of North American did very nicely in running both parts of the meeting when I was incapacitated. We made excellent progress in defining all of the various tasks in [sic] barring any difficulties within individual plants about getting men and money assigned to this work. Had a tape recorder at the meetings and we got most all of the important parts of the discussion on tape. It turned out the translation program actually could be made quite straight forward by stealing the symbolic section out of the SHARE assembly program and making the language entirely a function of punctuation. Convair, San Diego feels they can handle the job in its present form. M. I. T. will handle preprocessing for the control element, since that involves making up the pseudo instructions for control element which we must design anyhow. The other portions of the system appear to be in good hands and will be handled adequately. We established a system of semi-monthly reports to M. I. T. and to coordinating sponsors of the various tasks and M. I. T.

will issue monthly reports to the entire group and the A.I.A. as well. The first one is to be due for the Denver meeting. During the meeting we revised the radius of curvature calculation for the APT II system to correspond to that of APT III and also decided that a more sophisticated check service calculation is required. I have just developed a check surface calculation ..." And regarding the language itself: "Since we also left the Translation Program completely independent of spelling of words etc. we can throw the problem of machine tool programming language back into the laps of Benetar and Carlberg. I should do so in a letter describing the structure of the translation program" [R57520-24].

Since this paper concerns the history of the APT Language rather than of APT itself, I will not go into the structuring of the project or how the system and work was organized except when they influenced the syntax or semantics of the language. (Each company assigned six-character symbols beginning with their unique company letter, for interfaces, and each programmer specified his required inputs etc. [R574(sic)29, p3]). Starting with June 14, 1957, a series of "2D APT II" memoranda summarized both technical and project coordination matters in a series ending with my "Post Coordinators Report re Phase I System" of July 25, 1958.

6. PERIOD 4: THE ORIGINS OF APT LANGUAGE (May 24, 1957 to June 14, 1957)

I did the basic design of the APT language literally over one weekend -- not by design, but out of necessity. How that came about is as follows.

6.1 Prior Language Suggestions

I had, of course, the influence of Siegel's language, and our Special Course preparation discussions. But also, at the March 1 Programmer's Meeting, Boeing handed out a slightly updated description of their "Numerical Control Library Routines" [Boeing 1957b] (also presented in the Dallas Meeting) and also a one page "Part Programming Language Numerical Control Program - Tentative" (February 27, 1957)[Boeing 1957a]. Several of the APT words such as SETPT (set point), TRACO (Transform coordinates), ONKUL, OFKUL (coolant on and off), as well as XLARG, XSMAL (large and small X coordinates for multiple intersections) seem to have come from this list, for I do not remember these five-letter spellings being bandied about at MIT earlier. Also in preparation for the April 23-24 Programmer Meeting, I received a March 27 letter with attachments A through F from Ed Carlberg of Boeing [Carlberg 1957] stating "It should be our goal to present to the SNC on April 25 a united Steering Committee opinion and/or recommendation in the field of data processing compatibility, with the further goal of obtaining industry concurrence at the next Study Group meeting in May." Attachment A is a ten page paper on "Numerical Control Compatibility" in which Ed stresses compatibility with a minimum of standardization. He outlines 13 steps going from the engineering drawing to the machine tool and finds that the "manuscripts [will] allow the greatest flexibility in compatibility and will be the easiest item to exchange of any of the possible points" [p5]. He goes on:

> "Standardizing on the definitions is of much greater importance than on the actual word itself. If one plant calls it a 'curve' and another a 'circle', this means only minor re-programming or manual revisions to the manuscript in order to trade manuscripts. But if the definition is not standard, then a complete new manuscript starting from the part drawing is required. The number of words in a language will vary from plant to plant... Certainly all the SNC can do at this point is to establish a minimum language... [p6]. The other possible compatibility points (except the subroutine library) [have] unnecessary restrictions in the data processing without compensating benefits... [p7]. Thus there would seem to be no point in pursuing the study of compatibility points beyond the manuscripts" [p8].

He then goes on to emphasize also the need for subroutine "entry-exit (or front end)" compatibility so that "plants can interchange these detail subroutines, with only main control modifications required" [p9]. Interestingly enough, he finished "Of the two, the trading of parts is much more important from a long range, national defense standpoint. This compatibility should receive the bulk of our attention" [p10]. To this date, however, part program portability is just as difficult and rare as computer program portability. But as this shows, initially the industry did not desire the constraint of a standard language but wanted to achieve compatibility while retaining freedom to choose the words.

Attachments B and C provided definitions for the list of words referenced previously in the March 1 Programmer's Meeting, showing that the Boeing language was fixed-column card format, invoking program selection from the subroutine library approach. Items D and E then apply this to the preparation of a "manuscript" for the AIA Test Piece, an aluminum part with curves and pocketing which had earlier been established by the SNC. Attachment F presented Boeing subroutine standards for the IBM 704.

Among the assignments made at the Los Angeles meetings in April had been one to Vic Benetar of Lockheed, Georgia to propose the "standard manuscript language." On May 15, I received his initial recommendations for use in the meeting [Benetar 1957]. The list of 10 words had now grown to 17 including some which may have come from the Boeing list mentioned earlier. Vic also recommended 7 forms of circle definition, 10 forms of lines, 8 forms of points, and a general curve given by a list of points. The list of words is more closely related to Siegel's language than APT, however, for LINE and CIRCC were recommended for moving along a line or counterclockwise

around a circle, and as in Siegel's system all the symbolic names for circles began with C, lines began with S, and points began with P. Although I am sure this proposal must have been discussed somewhat, I cannot recall being influenced by it in my initial design of the APT language. I remember feeling that since Benetar's suggestions carried none of the flavor of our three-dimensional, more generalized work, I would have to suggest the basic language form myself if things were to match up right in the long run. Although I said in my Friday, May 24 resume "we can throw the problem of machine tool programming language back in the laps of Benetar and Carlberg. I should do so in a letter describing the structure of the translation program" [R57524], over the weekend I instead wrote my own language, on May 25 and 26, 1957.

6.2 The First Language Memo

Memorandum 2D APT II-2, dated June 14, 1957, is the first APT language definition. The memo is titled "A Proposed APT Language for the 2D APT II" by Douglas T. Ross, [signifying a two-dimensional, curve programming, APT system language][Ross 1957e]. My resumes indicate that it actually was primarily written over the weekend of May 25 and 26 (as I have described many times from memory over the intervening years). The resume for May 27 says

> "Started working out a language for the AIA APT II to suggest to Benetar on the basis of the results of our [kickoff] meeting. I have decided that we probably should have multiple words in the major word portion of the instructions and these sections will be separated by commas. In this way all of the instructions modifying the main instruction will appear on the same card. If not stated, they will remain the same as before. This can easily be programmed by having indicators set whenever a word occurs and each program will check with these indicators when appropriate. I have also decided not to store inside-outside information with surfaces for this system, since it is easier to merely say where the tool should go. In our final system we should have both possibilities. In a place where questions arise such as planes where it is not really sure which is the inside or the outside in the canonical form [this] can be handled again in terms of the normal. I suppose it could be handled, however, by saying whether you go through the surface or just up to the surface when you reach it. I will summarize these considerations in a letter and send it to Vic and suggest that the completion of the language be a function of the subcommittee and he with the help of Carlberg might consider themselves a project group for this task" [R57527].

I never followed through with the letter, since I did the memo instead. The reference to going "through the surface or just up to the surface" later became GO PAST and GO TO (along with GO ON) to control the check surface calculation.

Because of its historical importance, the 6-page initial language definition memorandum is included here in its entirety. The general types of APT language statements still apply today, with the use of modifier words separated from parameters by a slash and the incorporation of useful default cases, (which we always have preferred to call "normal" cases!)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Servomechanisms Laboratory
Cambridge 39, Massachusetts

Memorandum 2D APT II-2

SUBJECT: A Proposed Basic Language For The 2D APT II

FROM: Douglas T. Ross

DATE: June 14, 1957

The following pages present the format and vocabulary of a proposal language for the 2D APT II System. The format is defined by the types of statements which can be made in the language. Notice that the character space is completely ignored and that use is made of space in the vocabulary to provide easy readability. Since card format is to be variable field, the double dollar sign indicates the termination of a statement, and a complete statement need not appear on a single card. The form of instructions have been chosen for generality and for ease in converting into the coded form required for the initial control element program. Notice that the type of curve which is to be followed is derived from the definitions of the symbols used. The modifiers DNT CT and TERM permit symbolic positioning of a tool and establish meanings of later modifiers. For the initial system at least it will not be possible to use NEAR and FAR in definitions since the preprocessing programs will not have any means for determining the current output position so that the words NEAR and FAR would have no meaning. Note that it is possible to include a number which specifies feedrate in any statement immediately preceding the double dollar sign. In order to simplify the examination of the number of parameters for automatic selection of preprocessing routines it may be desirable to proceed feedrate by an additional "/" so that feedrate is uniquely shown. To assist in the interpretation of the language an example is given on the last page. The programming for the example would read in English as follows:

"Execute the following instructions at a feedrate of 80 inches per minute. From point P, don't record cutter coordinates, but go to point Q. With the tool on the left, don't record cutter coordinates, but go left on A to the near intersection of A and B. Then go left on B to C, go right on C to D, go left on D to E". (Notice that the near interesection has been used in all cases because if no specification of

NEAR or FAR is given, NEAR is assumed by definition). "Cross curve E and go along F to the far intersection of G. Then go clockwise on G to the near intersection of G and H, and counterclockwise on H to the (NEAR) intersection of H and I. Considering the tool to be on the right, terminate as though you were to go left on I, then stop. This is the end of the program".

LANGUAGE FORMAT

A. TYPES OF STATEMENTS

Definition

Symbol = Modifier,..., Modifier,
Name/Parameter,...,
Parameter $$

= , , , , / , , , , $$

Non-Definition

Symbol) Modifier, ..., Modifier,
Instruction / Parameter,..., Parameter
$$

) , , , , / , , , , $$

Parameter Definition

Symbol) Modifier,..., Modifier,
Parameter Name = Parameter,...,
Parameter $$

) , , , , = , , , , $$

Special Words

Symbol) Modifier,..., Modifier, Word
$$

) , , , , $$

Synonyms

Symbol = Parameter or Symbol $$

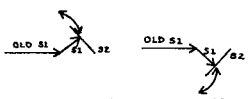
= $$

B. INSTRUCTIONS

Terminated by "," or "/"

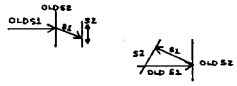
FROM S
Defines current cutter location S. S must be a point

GO TO S
Move cutter center to S. S must be a point



GO LFT S1, S2
GO RGT S1, S2
Go left or Right on curve S1 until S2 is reached.

(These instructions are usually used to go from one curve to the next. If only one symbol is given, it is interpreted as the new S2, and S1 is taken to be the previous S2.)



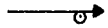
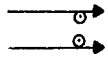
GO FWD ~~CROSS~~ S1, S2
Go BAK ~~BACK~~ S1, S2
[in original] Cross onto S1 and go forward until S2 is reached. Back up onto S1 and go in reverse direction until S2 is reached. (These instructions are used when curves are tangent or intersect at small angles. Two symbols are always required.)

GO CLW S1, S2
GO CCW S1, S2
Go clockwise or counterclockwise on S1 until S2 is reached. (These instructions are treated as the appropriate one of the previous four instructions.) If only one symbol is given it is interpreted as the new S2, and S1 is taken to be the previous S2 if GO LFT or GO RGT are appropriate; an alarm is given if CROSS or BACK are appropriate.)

PS IS S1
DS IS S1
CS IS S1
Part, driving, or check surface is S1

C. MODIFIERS

Terminated by "," or "/"

TL RGT 
TL LFT 
Cutter (tool) to right or left of curve when looking in direction of movement. These words also modify all following instructions.

DNT CT
Don't cut for this instruction only. (This modifier is used to prevent storage of cutter coordinates, etc., for the instruction which it modifies. The instruction, therefore, has the sole function of establishing cutter position and direction for later left, right, etc., instructions.)

TERM
Terminate the sequence of cutter motions as though this instruction were to be executed next. Applies only to this instruction. (This modifier is the converse to DNT CT since it defines the end position of the cutter. If only one symbol is used in the instruction being modified the modifier TERM causes it to be interpreted as S1 not as S2 as in the usual case.)

NEAR
FAR
Go on S1 until the near or far intersection of S1 and S2 is reached, with respect to

the specified direction of motion. "NEAR" will give the first intersection, "FAR" will give the second intersection, "FAR, FAR" will give the third intersection, etc. (If no specification of NEAR or FAR is given, NEAR is assumed. "NEAR, FAR" or "NEAR, NEAR" will give a translation alarm, but will be treated as NEAR alone.)

D. DEFINITION NAMES

Terminated by "," or "/"

CIRCL	Circle
ELIPS	Ellipse
PARAB	Parabola
HYPRB	Hyperbola
LINE	Line
POINT	Point
CURVE	Curve
INT OF	Intersection of
TAN TO	Tangent to
SPHER	Sphere
PLANE	Plane
QDRC	Quadric
SURFC	Surface
ZFNXY	Z = F(X, Y)
YFNX	Y = F(X)
CONE	Right circular cone
CYLNR	Right circular cylinder
CTR OF	Center of

Note: Each of these names has an associated canonical form, but preprocessing allows data to be given in other forms in many cases.

E. DEFINITION MODIFIERS

Terminated by "," or "/"

1
2
.
.
.

Integers tell which of several preprocessing programs to use with the name being modified. Not needed if data format distinguishes cases.

LARG X	(or Y or Z)
SMAL X	(or Y or Z)
POS X	(or Y or Z)
NEG X	(or Y or Z)

These modifiers are used with INT OF or TAN TO to tell which of multiple intersections or tangents to use. As many may be used as necessary.

F. PARAMETER NAMES

Terminated by "="

TOL N
Maximum allowable deviation from the defined curve = N

TL DIA N
TL RAD N
Diameter or radius of cutter. (For ball, or face or end mill)

CR RAD N
Corner radius of cutter

TL 1 (or 2 or ...) N1, N2, ...
N1, N2, ... are the parameters for a tool of type 1 (or 2 or ...)

FED RT N
Feedrate = N inches per minute.

G. SPECIAL WORDS

Terminated by "$$"

DNT CT
Don't cut any of the following instructions.

CUT
Cut all of the following instructions. (Used only to over-ride a previous DNT CT Special Word)

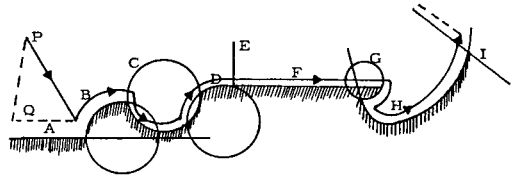
TL RGT
TL LFT
NEAR
FAR
Same as Modifiers, but apply to all following instructions (NEAR assumed if not specified)

STOP
Planned machine stop where all programmed motions are stopped, including the tape transport.

END
End of program or tape.

OTHER WORDS which depend on auxiliary functions of the particular machine tool being used.

H. EXAMPLE




```
FED RT = +80. $$
FROM / P $$
DNT CT, GO TO / Q $$
TL LFT, DNT CT, GO LFT, NEAR / A,B $$
GO LFT / B,C $$
GO RGT / C,D $$
GO LFT / D,E $$
FAR, CROSS / F,G, $$
NEAR,GO CLW / G,H $$
GO CCW / H,I $$
TL RGT,TERM,GO LFT / I $$
STOP $$
END $$
```

NOTE: ANY INSTRUCTION HERE CAN HAVE FEEDRATE GIVEN BEFORE "$$"

IF ONLY ONE SYMBOL IS USED IT IS THE DESTINATION CURVE (EXCEPT FOR "TERM") I.E. COULD HAVE

```
"- - - - - - - - - - - - - - - -
GO LFT / B,C $$
GO RGT / D $$
- - - - - - - - - - - - - - - - "
```

INSTEAD OF

```
"- - - - - - - - - - - - - - - -
GO LFT / B,C $$
GO RGT / C,D $$
- - - - - - - - - - - - - - - - "
```

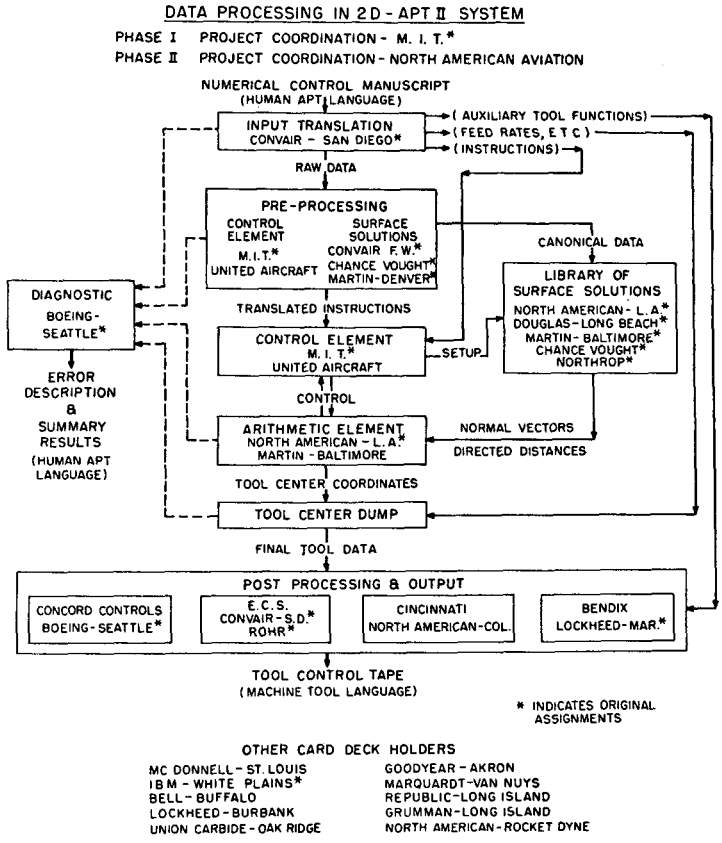


Figure 3 The 2D-APT II Joint Programming Effort

So it literally is true that I designed the APT language over a weekend and that its principal syntactic features were determined by the requirement that its translation be controlled entirely by the statement format punctuation. The basic simplicity of the core of the language has made it easy for part programmers to learn. As we will see, later difficulties with the calculating portions of ARELEM, as well as the more sophisticated requirements of elaborate geometric definitions and transformations, enriched the language considerably (and made certain portions more of a challenge to use correctly) but the basic sense of the language has stood the test of time well.

7. PERIOD 5: EVOLUTION OF THE SYSTEM (May 1957 through May 1958)

7.1 Project Organization

The Language Memo was written immediately following the Kickoff Meeting as the project was just getting underway. The first MIT Coordinator's Report was issued by Clements June 21, 1957 [Clements 1957], and reports progress in some areas, and not in others. There is neither time nor space to go into details here, but Figure 3, the Press Conference version of a diagram extracted from the first paper on APT [Ross 1957d] (written between September 13 and 27, and presented October 24, 1957, at the Third Annual Contour Machining Conference in Los Angeles) shows both the system structure and the company assignments, originally and in February 1959. The key elements to note concerning the language aspects are that Input Translation (INTRAN), Definition Preprocessing (DEFPRE), and Instruction Preprocessing (INSPRE) constitute the translator proper. They served to compile interpretive code for CONTROL and ARELEM, in which resided the real semantics of the language. ARELEM turned out a cutter location tape (CLTAPE) format to a changeable Postprocessing and Output program. Machine-tool-dependent vocabulary was passed directly through, in sequence, for handling by the chosen postprocessor. A library of Unit Normal and Directed Distance subroutines, with one canonical form for the data of each surface type, and the Diagnostic system completed the system architecture still followed pretty much today. These company assignments realized the modularity cited in premises 2 and 3 of my theme (Section 1.1).

7.2 Systematized Solutions

The basic idea that a systematized solution provides the semantic framework for a specialized application language, thereby permitting an integrated, cohesive approach still seems not to be widely recognized even in the nineteen seventies, although it is in many ways the essence of any strong argumentation based on abstract data types. Twenty years ago the concept really was difficult for people to grasp, even with our simulated computer analogy.

The question of systematized solution was considered of such importance that the majority of Chapter 1 in Interim Report Number Three, covering the period from January 1 to March 31, 1957 [MIT 1957a], is devoted to illustrating the principle by generalizing a program to find the intersection of a parabola with a circle (from the Special Course) to a program to find the intersection between two arbitrary plane curves.

> "Just as the example of the previous section solved for the intersection of any two curves providing the defining functions were known, the two systems to be described [for APT II and APT III] move a cutter along a space curve or over an entire region, providing the analagous defining quantities are available ... As [is] described more fully in the last chapter of this report, it is planned to attempt the same type of systematized solution to the problem of translation of languages, so that the translation portions of the APT computer can have the same generality and flexibility as the calculating portions of the computer. The resulting systematized approach to the problems of APT system design will, it is hoped, result in a basic structure and an accompanying methodology which will allow APT systems to continue to grow and change to meet the varying needs of their users" [p13].

> In fact, the last chapter of that report concludes with "At the present time, it seems possible that the two research efforts outlined above -- determining the principles of language design and language translation -- may provide sufficient cross fertilization to make it possible to devise a special metalanguage which can be used to describe special-purpose languages of the type under consideration. If this is indeed possible, then it may prove feasible and desirable to devise a program to write translation programs based upon a description in the metalanguage, of the rules and vocabulary of the subject language. Such a program would represent an important advance in APT system design, since the individual programmer could add to the language of the system at will" [p52].

According to my resume that was probably written May 15, 1957, a good four years before I developed my Algorithmic Theory of Language [Ross 1962], and seven years before the initial version of my AED JR system, [Ross 1964] which actually did what was suggested here. The prospect was very real to me at the time, however, and I suspect that it was this initial connection between systematized solution and meta-language which caused me to emphasize in the Algorithmic Theory of Language that language design was primarily semantic rather than syntactic. This also was the reason why when we finally did develop the AED language we did not go beyond AED-0 [Ross, Rodriguez, and Feldmann 1970], and instead extended the language by "integrated packages" of functions (corresponding to the abstract data types of today) so that the language was extended semantically without syntactic extensions. This enabled AED to possess in the middle 1960s, features that other high level languages only achieved many years later.

As we will see, even though the tremendous turmoil of practicalities of APT system development caused many of these rich concepts to be set aside initially, some of them did come to the surface and had an effect later in the Definition Preprocessing and Macro features, which were my last contributions to APT language development in 1960, predating similar features in other high level languages.

7.3 Evolution of the Translator

As I have mentioned, the starting idea for the Input Translator (including the funny punctuation) came out of the May 23, 1957 session of the Kickoff Meeting. "The translation program actually could be made quite straight forward by stealing the symbolic section out of the Share assembly program and making the language entirely a function of punctuation. Convair, San Diego feels they can handle the job in its present form" [R57520-24]. But by June 10 Convair dropped out of the project [Kinney 1957], and not until a month later were they cajoled to return. Finally, Charlie Swift did visit with me at MIT for the whole day of August 14 [R57814] at which time we worked out the basic translation scheme involving a data table filled from card reading, a syllable table for words, a symbol table, and communication to the preprocessing programs -- all controlled by the statement punctuation.

Things were progressing very erratically in the widely dispersed project, so by August 30 I joined Don Clements in preparing the Coordinator's Reports. The "new style" (i.e., no-holds-barred commentary on any and all aspects of the project) Coordinator's Report [Ross and Clements 1957] begins to show further evolution of the language ideas. The discussion of the input translation process shows that it was what would today be called a "lexical scanner".

The fact that the APT language at this point in time had only four statement types which were determined entirely by their punctuation had a profound influence on the evolution of the language at this early stage. The September 27 Coordinator's Report discusses various modifications to the input translation process to allow transient modifiers to both precede and follow words which they modify, and in the next Coordinator's Report, October 22nd, I made what became the final design evolution for this early stage of INTRAN by suggesting that a table for modifiers similar to that for parameters should be used so that the sequence of words would be passed along intact to INSPRE and DEFPRE. This, in effect made the input translation program into a proper lexical scanner which merely recognized, translated into internal form, and type-classified the words in the sequence they were encountered in the input string. This greatly simplified the scheme and I said "This rather major change [may make Convair] feel that their task is becoming too simple and that their talent is not being made use of" [Ross and Clements - 1957 October 22]. Interestingly enough, on October 31 (after the above simplifications) I received a copy of an

October 24th letter from Charlie Swift to Ed Carlberg [Swift 1957] saying that it was useless for the Convair programmers to attempt to continue their task until the language was more pinned down. Therefore, the suggestion when it was received, was well received.

The final change to the translator structure which formed the framework for further language design developments also was recommended in the October 22 Coordinator's Report. This consisted of the specification that INSPRE should be allowed to call DEFPRE as a subroutine. The Instruction Preprocessor handled all words to the left of the slash (/) while the Definition Preprocessor was invoked only whenever there was an equal sign (=) in column seven. By allowing INSPRE to call DEFPRE, the language was considerably enriched and made much closer to my original desires for the language because the programmer now could write, for example, "TL LFT, GO RGT, LINE/P1, P2 $$", i.e., geometric definitions did not have to be given separate symbolic names but could be nested in the motion instruction itself.

The final clincher to the translation scheme is contained in the following letter of November 27th [Ross 1957e] to B. J. McWhorter of Convair which I quote in its entirety to give a first hand feel for both the methods we were using and the flavor of the informal communication which, in those days, took the place of "formal specifications":

> "We just received your progress report of October 20 and although I have not given it very careful perusal, there are a couple of things I thought I would bring to your attention and see if you can get them into the deck that you will send us by December 1. The change is slight and comes about because of the fact that we plan to handle surface definitions as instructions in Phase I, and also since your progress report does not seem to have quite as much flexibility as you said you needed when you were here. The change is, therefore, that the modifiers on page 7 of your report, XLRG, etc., should go to the right of the major punctuation, / , as well as the TAN TO etc., modifiers on page 8. In other words, all of your important modifiers will appear in the translated parameter table. I would suggest that the coded form for TAN TO etc., be changed to read (1, 0, 2) etc., when coded in SAP. Or if you do not need to distinguish the two classes of modifiers, then just number those on page 8 from 9 through 14. These coded forms will then all appear in the symbol table and not in the syllable table which is reserved for words to the left of the major punctuation. Your example, then, would be changed to read as follows:
>
> LIN 1 = LINE, 3/ LARGE Y, TAN TO,
> CIR 1, SMALL Y, TAN TO,
> CIR 2 $$
>
> "I should also point out to you that the geometric type of the figure which appears in the decrement of symbol table entries as you give them on page 7 will only use the right hand 6 bits of the decrement. You should mask out these 6 bits since we may very soon have junk in the left hand half of the decrement. If you can describe any of the special cases for which you want additional information in the decrement in a rough way, please let us know since we are at present planning to assign these additional positions on the basis of control element requirements.
>
> "One further thing, is that type 1 modifiers, i.e., numbers, still may appear in the translated modifier table as modifiers to the principal word, in case you need this information for selecting among preprocessing routines. If control is passed to your program by the instruction preprocessor, then we would prefer to have the numerical modifier follow the principal word, i.e., line, 3 instead of 3, line in your example. Since we will have recognized the word LINE as calling for your preprocessor, we can still pass that along to you as the principal word in which case the 3 will be the last entry in the translated modifier table.
>
> "I hope this letter is not too hastily written and that you can make sense out of it. The new language memo is almost completed and so far I plan to put in only canonical forms for surface definitions but if you have any additional suggestions to pass along at this time please do so immediately. We are looking forward to getting a deck of cards from you soon for Phase I. Hope you had a happy Thanksgiving."

With these changes, the evolutionary pathway for the language enrichment was established. From this point on, the interactions of the semantics of the language, in terms of our ability to carry out the geometric computations, could be reflected directly in a simple choice of a new syntactic word and the corresponding control element action. Even though there was no formal meta-language definition, the (now) well-understood relationship between the Input Translation lexical scanner, INSPRE, and DEFPRE, and the Control Element program and ARELEM provided, in an informal way, the mechanism for a fairly clean enrichment of the language (taking the place of a formal language theory). The next stages of evolution had to do with the semantic vocabulary areas -- a process that, due to the complexity of both the subject matter and the project organization, was to continue for another three years.

7.4 The ARELEM Problem

My primary contention in this paper is that it is orders of magnitude more difficult to create truly new, deep, application-oriented

languages such as APT. In order to be successful, they must be based upon a world view that both matches the thought processes of the intended end users and also is amenable to efficient, reliable, and comprehensive computations. Only if the coverage of the world view is complete, comprehensive, and reliable can the user himself think completely, comprehensively, and reliably in terms of the language. This is why throughout the actual historical development of APT as well as throughout this paper, I have repeatedly returned to the theme of the "systematized solution", for it embodies this abstraction of the essence of the entire problem area in generic form so that each specific problem that is posed in the language in fact has a workable solution. In the case of APT, this required the establishment of a methodology for solving the dynamic three-dimensional geometric problem of approximating arbitrarily closely, the sculpturing of arbitrary three dimensional parts composed of arbitrary surfaces by means of broken-line motions of rotating tools, described by arbitrary surfaces of revolution, with five degrees of freedom -- three of position and two of tilt.

To appreciate the intellectual difficulty of the task you must realize that the "mind's eye view" of ARELEM is completely different from that of a person considering the problem. A person can appreciate the fact that the task is difficult if he is told that the problem is to consider an arbitrary-shaped tool (a surface of revolution) which can be steered through space, and his problem is to do that steering in such a way as to cut an arbitrary-shaped part out of a chunk of material. Both the speed and direction of approach, as well as the actual point of contact, must be very precise so that there is no chatter or gouging, etc. Clearly the problem is non-trivial.

With the concept of arbitrary shapes somehow floating and moving in space in a precise and coordinated fashion firmly in mind, now consider the problem once again from the "mind's eye view" of the actual ARELEM computation. In this case, there is no direct concept of shapes, surfaces, and smoothness of surfaces and motions. Instead, there are only sets of three-dimensional coordinates making sharp angular constructions of vectors, vectors, and more vectors. Everything is extremely discrete, jumpy, and disjointed. There is only a staggeringly complex construction of points and straight lines connecting those points. Even such a simple concept as curvature (a small segment of a circular arc in some oriented plane, with a point in the center of that arc) is not directly known but must be approximated by an appropriate construction which includes

1) the selection of the orientation of the plane specifying the direction in which curvature is to be measured,

2) the selection of some appropriate base point from which two directions can somehow be selected

3) from which two applications of the directed distance subroutine for the surface can be invoked to provide

4) two points exactly on the surface in question

5) at which the unit surface normal vectors can exactly be calculated

6) from which suitable projections into the plane can be made to allow

7) the construction of two circle radii and the chord between them so that

8) the curvature can be estimated.

(And throughout the process, remember that the directions for the directed distances must yield points on the curve of intersection between the plane and the unknown surface, if the whole approximation process itself is to have any validity.) So it takes all that merely to approximate a single curvature in a single direction. From this horrible description of the simplest of all ARELEM operations you can perhaps appreciate the difference between the human and ARELEM viewpoints. Points and vectors between points (relations between points) -- that is all ARELEM has to go on. That is all that unit normals and directed distances supply. There are no analytic functions known to ARELEM, for whatever their form, they would limit its generality and applicability at its most basic level. The APT world is primitive indeed.

This paper is about language design, and this brief belaboring of the computational difficulties is included only to pin down the fact that ultimately every aspect of the APT language must be reduced to these same, very primitive terms. Furthermore, when, because of the mathematical vector constructions within ARELEM, multiple cases must be selected from, words with meanings suitable to the mind of the part programmer must be included within the language so that he can supply the appropriate constraint to the degrees of freedom of the vector computation. Several APT words arose from this source.

7.5 Evolution of ARELEM

It should not be surprising that the solution to this problem required several years, and that the viewpoint and terminology of the APT language went through an evolutionary development during this period, corresponding to the successes and numerous failures of ARELEM calculations.

Throughout the early period from 1957 to 1959, the primary difficulties driving this evolution hinged on the gradual weaning from the holdovers of the methods that were successful in the much simpler two-dimensional case in which tool center offsets are applied separately after the approximation to the desired curve has been established. In the case of two dimensions, even for arbitrary curves, tool center offset requires merely the expansion of the curve in the direction of the surface normal the fixed amount of a tool radius. To be sure, such addition of a normal direction thickness to a convex curve can lead to cusps and other difficulties, but except for the detection of these anomalies (and the corresponding

problem that, of course, an inside curve can have no sharper radius than the tool radius) the problem is quite tractable. Therefore, a completely workable, comprehensive, and reliable two-dimensional system can be created with this underlying viewpoint of a zero-radius "point cutter" followed by a separate "tool center offset" computation.

Even though the initial AIA joint effort APT system was to be "2D APT II" this always was recognized as a mere waystop, and from the beginning ARELEM was to be three-dimensional [Ross 1957b]. The difficulty was that we did not realize at that time that the concept of a separate tool center offset for three dimensions is essentially unworkable. Furthermore, the urge to creativity, coupled with the fact that we had only incomplete documentation on our Whirlwind APT II programs [MIT 1957a, R57122, 3] then being debugged, caused O. Dale Smith, of North American Aviation (NAA) to make the first IBM 704 ARELEM with a new and different set of equations [Smith and Corley]. This both introduced new problems and made it impossible to exchange results between the two efforts -- both of which suffered from the deadly "point cutter with cutter center offset" syndrome! Thus there was a long period of "improvements" in the basic ARELEM computation in which case after case of anomalous behavior was corrected for by the application of special treatments [R580114, AIA 1959]. Interestingly enough, most of the linguistic aspects of these problems were of a general enough nature to last. Even when I finally devised the successful MIT ARELEM (with the assistance of Clarence Feldmann, see Section 12, from late 1959 through 1961), both the situations and viewpoints for the proper application of the words FROM, INDIR, MAXDP, and TNCKPT remained valid. Quite probably most current users of the language are unaware that those terms entered the thought framework of APT at such an early and rudimentary stage.

7.6 Evolution of Semantics

My voluminous records would permit almost a blow by blow account of the evolution of the meanings of almost every APT language word and construct. But suffice it to say that with much travail, and many Coordinator's Report exhortations, progress was laboriously made on all fronts during the summer and fall of 1957 [Ross and Clements 1957]. At the October 3 and 4 meeting of the AIA Subcommittee for Numerical Control I requested that the cooperating companies "take a relatively simple part that they plan to produce by numerically controlled machines and try to program it by using the language which was supplied in the 2D APT-II Memorandum 2, and to send the result to MIT by November 1. This would show up any inadequacies in the current language proposal and enable MIT to provide a more usable language" [AIA 1957c, p4]. Also at that meeting it was decided that since the October date could not be met, the new target would be a January 1, 1958 distribution of a system, if the companies sent their program decks to MIT by December 1. As it turned out, this date, too, could not be met, for the last deck reached MIT February 3, 1958 [R5823], and the February 4 Coordinator's Report states "It is hoped that a working system can be performing before the next SNC Meeting on February 20." The March 13 Coordinator's Report says

> "At the (SNC] meeting it was reported that a successful part program had been run using the 2D-APT II program and that the debugging operation was still in progress... On this note, SNC arranged [that] a 'field trial' distribution of the program was to be made in the early part of March as a sort of shakedown before a formal distribution of the program... The target date for the distribution of the more finished version of Phase I was selected as the first of May. During the interval between the March distribution and the May distribution, MIT agreed to produce an edited and comprehensive version of the documentation and to furnish a corrected card deck to the APT Project Coordinating Group [headed by O. D. Smith of North American] for distribution to the industry. There will be included in the May distribution, three items: A part programmer's manual, a computer programmer's manual and descriptive brochure. This will complete MIT's formal participation in the joint effort. After May, the MIT project will revert to a consultant status and concentrate on further research efforts for AMC [Air Materiel Command]" [Ross and Clements 1957].

Thus was the Field Trial first version of APT defined. We now move back again in time to pick up the language definition thread that led to the corresponding Field Trial Part Programmer's Manual.

8. PERIOD 6: THE FIELD TRIAL APT LANGUAGE (November 1957 through April 1958)

8.1 Two Papers on APT

To set the background, on August 15, 1957 I had received an invitation from George Fry, President of TruTrace (who had given the demonstration prior to the AIA decision to start APT) to present a paper at the Third Annual Contour Machining Conference to be held at Los Angeles October 23 through 25. Since this would be the first chance to describe APT and the AIA Joint Effort, in a public forum, I accepted [C57819]. The paper was actually written sometime between September 13 and 27, disrupted by various activities including a week long debugging trip to Eglin A.F.B. [R57913-927]. Figure 3 from that paper [Ross 1957d] has already been presented. Figures 4 and 5, also from that paper, show Siegel language and APT language versions of a sample part program. At the time, this was the most complete part programming example I had written since the original June 14 Language Memo [Ross 1957c]. Compare the placement of the modifiers with the sample line in the letter to McWhorter (Section 7.3). (Notice that the October 22 Coordinator's Report with the improved translation scheme was completed just before I departed for this Los Angeles paper presentation.)

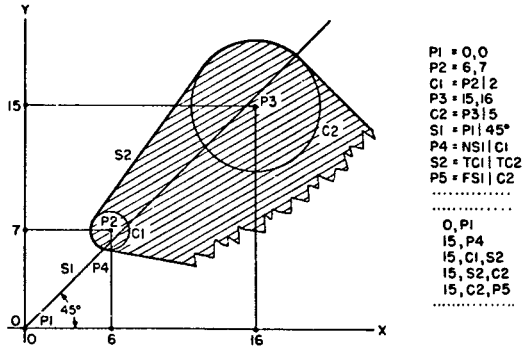


Figure 4 Sample Part Program, Siegel Language

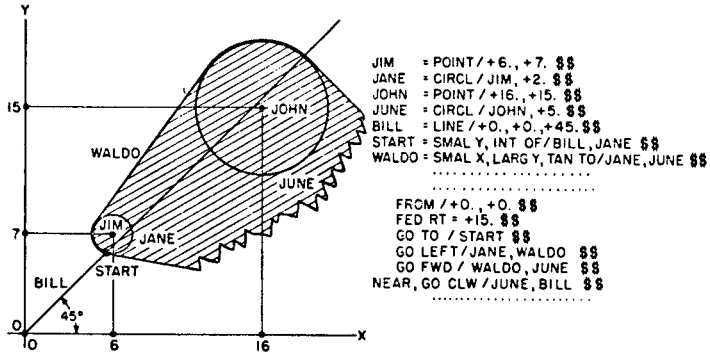


Figure 5 Sample Part Program, 2D-APT-II Language

Upon my return from Los Angeles, the November 8 resume says "We have got some sample parts programmed in APT language from some of the companies, some very good and some really sad" [R57118, p2]. So I also had those examples with their high incidence of tangent check surfaces as stimulation. According to the resume of November 12 through 18,

> "Harry Pople has returned from his stint in the Air Force ... I also described to him briefly the use of the Scope Input Program as a possible flexible input language for APT programming. He seemed quite taken with the idea. We will save this for a little bit in the future, however, and only try to do a very pilot model to begin with. ... I will start working on the draft of the language memo today [R571112-18].

As I mentioned earlier (Section 2), I had written the first computer graphics input program in 1954, and here I was suggesting its use for part programming, even though it was to come only many years later.

The November draft of the APT language memo (which I have) [N5711?] never was completed because it was interrupted by the activities of other projects, but in particular, another paper-writing stint. On October 7 I received an invitation to present a paper on APT at the ACM-sponsored session at the annual meeting of the American Association for the Advancement of Science in Indianapolis in late December [C57107]. On October 9 I was further informed that "The paper you are to present ... has been selected by the Association's Public Information Committee as one which should be reported widely to the general public. For your own protection I trust that you will cooperate with us ..." and requesting 100 copies of a non-technical abstract of the paper by December 15 [C57109].

I started thinking about the paper December 2, sent the abstracts in on December 12, and "Most of the time before Christmas was spent in writing the AAAS paper and taking a week's vacation from December 23 on. The paper finally came out quite well and was well received at the meeting" [R57123-58019]. It made the front page of the Indianapolis Star as I got on the airplane

Sunday morning, December 29, and upstaged Crawford Greenwalt 15 column inches to two in the next day's Wall Street Journal, etc. [R57123-58019]. But the paper was later rejected for publication in the Communications of the ACM. "While very interesting [it] is too highly expository for publication ... The Communications would, however, be interested in publishing a description of the programming language and a small example of its use. The philosophy of program construction, while interesting, parallels that in any other development of a specialized programming language. What would really be worth while would be a discussion of the steps by which a language for your particular class of problems has been developed" [Perlis 1958]. I certainly did not agree with this put-down and was still in the throes of language design and therefore I made no further efforts with ACM. Perhaps this paper twenty years later will fill the bill. Throughout my career editors and reviewers have reacted poorly to the same work that others (often years later) give me awards or accolades for.

The emphasis in the AAAS paper [Ross 1957f, 1958a] was on the "systematized solution [which] greatly clarifies the obscure problems which arise in spatial geometry, language design, language translation, and system organization. It is not improbable that continued work along these lines can lead to a design machine which will assist in the design process itself and then automatically produce a part to meet the specified conditions" [p16]. Although the language aspect is only mentioned in the AAAS paper, I illustrated for the first time, the idea of "ignorable words" with the sample statement "GO RGT, WITH, TL FT, ON, CIRCLE/CENTER, PNT 3A, RADIUS, +5.025 $$" [p26]. I also discuss some of the ground rules for "applying the principles of systematized solution to the problem of language translation" [p28]. My yellow-page hand written notes for the paper are of interest to me because they say "Man is programmed by the language we design for him to use, since only way he can get the system to perform is to express his wishes in the specified language form" [N5712?, p7], which is the earliest trace I have of my stock statement that a language designer bears a heavy burden of responsibility because he provides the major part of the solution of any problem that ever will be solved with his language because he programs (constrains) the thought processes of the human who will use the language to solve a problem.

In any case, these were the sources of background leading up to the April, 1958 Field Trial Part Programmers Manual for APT. The original June 1957 language memo, the clarification of semantic content brought out by refinements of input translation, instruction and definition preprocessing, and control and ARELEM interaction, along with the sample part programs of the Contour Machining Conference paper, test programs from the companies, philosophically-based first draft attempt at a field trial language memo, and finally the AAAS paper all taken together constituted the background source for the first true APT Part Programmer's Manual embodying the definition of a complete APT language.

8.2 The Field Trial Language

Whereas the June 14, 1957 2D APT II-2 Language Memo [Ross 1957c] consisted of six pages, the April 29, 1958 2D-APT II Part Programmer's Manual (Field Trial Version) [Ross 1958e] was 50 pages long. The flavor of the language at that stage is concisely conveyed by the now-famous "MIT Teardrop" sample part, see Figure 6, (a combination of Figures 1, 2, 3 from the Manual). Page 21 also, however, illustrates the more complex nesting of a geometric definition in a complex motion instruction, with post-processor control, by

TL LFT, ON KUL, FAR, 2, GO FWD,
ELIPS/+4.5, +5., ... $$

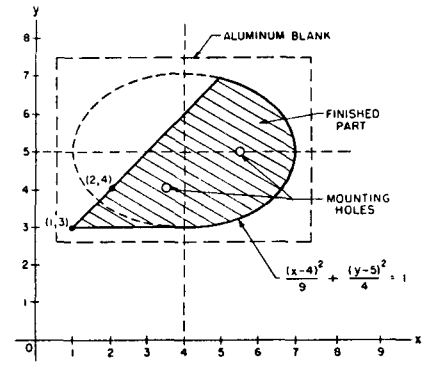
But page 36 points out that in this, more complex form, "feedrate may not be added before the $$", whereas it could be in the simpler forms. (Several other restrictions evidence various growing pains in the coordination between the program modules, in this case, between INSPRE, DEFPRE, and CONTROL. Once INSPRE passed control to DEFPRE, it could not send feedrate to CONTROL, as it usually did, and DEFPRE would only return control, not the value of feedrate it could have found!)

Examination of Figure 6 also shows that at this point the flat plane of the two-dimensional part was considered the Drive Surface (DS), whereas later this was made the Part Surface (PS). The example is missing a crucial comma in line 17, but neatly side-steps the startup problem covered in detail on pages 28 and 29 of the manual, where a sequence of three DNT CT, GO TO instructions demonstrates how "Illegal Use of Vertical Motion" can be avoided by a slanting dodge seen by ARELEM, but resulting in a simple vertical plunge by the machine tool. Since sense of direction came from comparison of the most recent motion with the direction of the tool axis, a plunge was impossible for ARELEM!

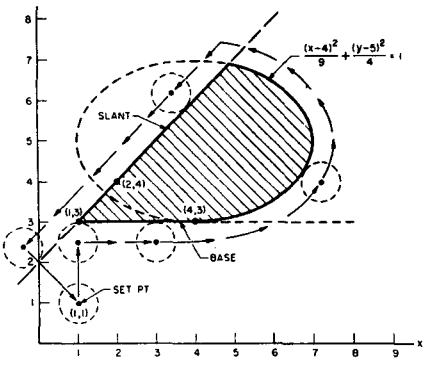
9. PERIOD 7: THE POSTCOORDINATOR REPORT (July 1958)

It was a lot of work putting together the first APT Field Trial System. Clarence Feldmann, who was to work closely with me for the next twenty years, had joined the project as a Research Assistant, June 28, 1957 [R57628] and had been responsible for MIT's INSPRE and CONTROL modules, as he rapidly learned the ropes. He, Harry Pople, Jerry Wenker, and Bob Johnson, loaned to the project by IBM, were the prime movers, although a week's visit to MIT by Dale Smith in February [R58210-17] also helped greatly with ARELEM integration, at the start.

The April 15, 1958 Coordinator's Report [Ross and Clements 1958] says "Accompanying this memorandum are the rest of the computer program writeups for the field trial of the 2D-APT II Phase 1 system... The IBM Service Bureau, shipped the field trial card decks [some 8000 cards



GIVEN DATA FOR SAMPLE PART



SYMBOLS AND TOOL PATH

Line Number	Portion of APT Program	APT Statements	Type of Statement
1	GEOMETRIC DEFINITIONS	DS = PLANE/+0.,+0.,+1.,+3. $$	Definition
2		SET PT = POINT/+1.,+1.,+3. $$	"
3		Y AXIS = LINE/+0.,+0.,+3,+0.,+1.,+.3.$$	"
4		BASE = LINE/+1.,+3.,+3.,+4.,+3.,+3.$$	"
5		CONIC = ELIPS/+4.,+5.,+3.,+.33333, +0.,+0.,+0.,+.5,+0. $$	"
6		SLANT = LINE/+1.,+3.,+3.,+2.,+4.,+3. $$	"
7	PARAMETER DEFINITIONS	TL DIA/ +1.0 $$	Instruction
8		TOLER/ +.005 $$	"
9		FEDRT/ +80. $$	"
10	TOOL CONTROL INSTRUCTIONS	HEAD 1 $$	Special
11		MODE 1 $$	"
12		ON SPN $$	"
13		ON KUL $$	"
14	STARTING INSTRUCTIONS	DS IS / DS $$	Instruction
15		FROM/ SET PT $$	"
16		GO TO/ +0.,+1.,+3. $$	"
17		TL ON GO RGT/ Y AXIS, BASE $$	"
18	CUTTING INSTRUCTIONS	TL RGT, GO RGT/ BASE $$	"
19		GO FWD/ CONIC $$	"
20		GO LFT/ SLANT, BASE $$	"
21	CUT FINISH INSTRUCTIONS	TERM, GO LFT/ BASE $$	"
22		GO TO/ SET PT $$	"
23	PROGRAM FINISH INSTRUCTIONS	OF KUL $$	Special
24		OF SPN $$	"
25		END $$	"
26		FINI $$	"

1-4

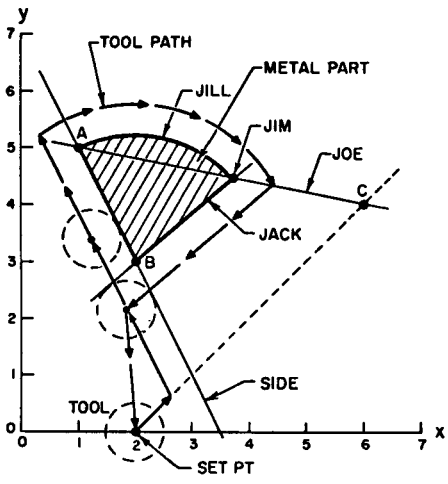
Figure 6 "MIT Tear Drop" Sample Part

to 16 companies] via Railway Express on April 10, 1958." Finally, the May 1 Coordinator's Report says that "The programmer's memo of instructions for the field trial version of the 2D-APT II Phase 1 system was mailed on April 30. On May 6, 1958 another mailing will occur which will contain a deck of cards for the sample part program described in detail in the report, instructions for running this test part through the system and up-to-date deck corrections. It is hoped that this manual and this test part will allow the recipient companies to start actual field trial tests for the system on their own." From that point on, the official debugging of the APT System was intended to take place in industry with coordination by Dale Smith of North American.

Although ostensibly MIT was to concentrate on the documentation, following the distribution of the APT Field Trial decks, we also continued with full-scale development and debugging runs on APT II and APT III on Whirlwind, along with intensive debugging of the 704 APT system. United Aircraft had joined the AIA Joint Effort, and by March 4, Dale Smith suggested that they take over the Instruction Preprocessor and Control Element responsibility from MIT [Clements 1958]. By this time also Sam Matsa, now at IBM, had obtained a go-ahead to develop an APT III region programming system for the IBM 704 [R58312-17], and United Aircraft was going to collaborate with IBM on this effort, which shortly was christened AUTOPROMT (for automatic programming of machine tools). It was decided that IBM would concentrate on the region programming aspects, while United Aircraft concerned itself with the incorporation of 2D APT II for the curve programming necessary for initial passes and treatment of region boundaries.

After various interchanges, a fruitful meeting between MIT and United Aircraft programmers took place at MIT on July 2, 1958. Ken Kaynor and Richard Leavitt met with myself and Clarence Feldmann, and a number of syntactic and semantic language issues were finally settled [Kaynor 1958].

There was a meeting of the Numerical Control Panel (the new name of the SNC) in Seattle July 16 through 18 at which both Dale and I made presentations on the status of the Field Trial distribution and rework. A polling of the companies represented showed that great difficulties were being encountered, but that most were still counting on APT being successful and switching to it once it was [AIA 1958]. I also met with the people at Boeing "going over the greatest parts of the Phase I system which is now defined as that system which MIT will now write up. Discussion was




```
A     = POINT/ 1, 5
B     = POINT/ 2, 3
C     = POINT/ 6, 4
        TL DIA/ +1.0, INCH
        FEDRAT/ 30, IPM
SET PT = FROM, POINT/ 2, 0
        IN DIR, POINT/ C
SIDE  = GO TO, LINE / THRU, A, AND, B
        WITH, TL LFT, GO LFT, ALONG/ SIDE
JILL  = GO RGT, ALONG, CIRCLE/ WITH, CTR AT, B, THRU, A
JOE   = LINE/ THRU, A, AND, C
JIM   = POINT/ X LARGE, INT OF, JOE, WITH, JILL
JACK  = LINE / THRU, JIM, AND, B
        GO RGT, ALONG/ JACK, UNTIL, TOOL, PAST, SIDE
        GO TO/ SET PT
        STOP, END, FINI
```

Care To Match Wits With A Giant Brain?

Marvelous new APT System, using large computers and automatic machine tools lets you cut out complex metal parts just by writing down the instructions. Read the "APT language" part program above, comparing with the picture, and see how brainy these brains are getting to be. Abbreviations are TL = tool, DIA = diameter, FEDRAT = feedrate, IN DIR = in direction of, LFT = left, RGT = right, CTR AT = center at, INT OF = intersection of. MIT scientists working for Air Force developed new Automatically Programmed Tool (APT) system with cooperation of Aircraft Industries Association companies.

Figure 7 Sample Part Program

concerned primarily with post processing and the translation and preprocessing" [R5814-21].

Upon returning to MIT, on Tuesday, July 22, I "Had a long go around with Harry and Clare concerning the shortcomings of the APT II system and the APT documentation... After much hashing around, we finally decided that we would issue a postcoordinators report describing all the changes that are necessary to make the field trial version into a Phase I system... and at the same time request changes in the SHARE writeups" [R58722].

On July 24, "Don and I called Dale Smith and although he seemed somewhat taken aback at first, after we explained what we meant by a post-coordinators report he seemed to think it was a good idea" [R58724]. The report itself was issued July 25 [Ross 1958f] and, in effect, it constituted the specifications for the program changes necessary to allow me to actually get down to work completing the Part Programmer's Manual.

The remaining portions of the crystalization of APT language ideas took place in my writing, correspondence, and discussion over the next few months. By November 14 "Drafting and typing are well underway" [R581113 + 14], but it was not actually received back from the printers for mailing until mid-February 1959 [R59210-18]!

10. PERIOD 8: THE FIRST REAL APT LANGUAGE (August 1958 through November 1958)

10.1 The Phase I Part Programmer's Manual

For all the ups and downs in deciding how to handle the syntactic and semantic processing, this first really complete version of the APT language [Ross 1959a] ended up with a number of very interesting features which not only strongly influenced the subsequent development of APT, but were strongly influencial on my thinking in the design of the phrase substitution and object-oriented-language features of AED. They do come quite close "except for the funny punctuation" to the objective of having APT be an "English-like" language. The Press Conference handout example, Figure 7, illustrates this best.

By the time the instruction preprocessor was allowed to handle all translation and call on the definition preprocessor when necessary, and when the same words, such as TO, ON, PAST, were allowed to appear either as modifiers in the major section or as parameters in the minor section, very English-like idea flow was made possible, especially if ignorable words were used, too.

The manual states [p1-7],

> "There are four types of APT statements containing one, two, or three principle sections, separated by distinguishing punctuation characters". These are:
>
> Definition statement
>
> Symbol = major section/minor section
>
> Instruction statement
>
> Major section/minor section
>
> Synonym statement
>
> Symbol = minor section

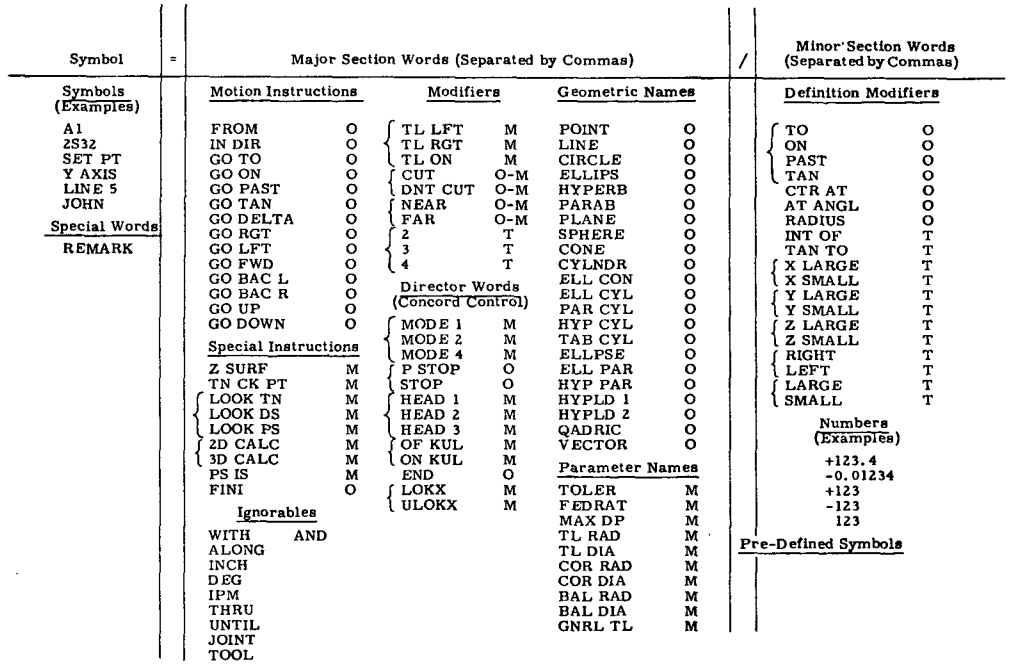


Figure 8 - Complete Initial Vocabulary for the APT II System

Special statement

Major section

"Each word exhibits a certain modifying power with respect to the overall meaning of the entire part program... called Modal (M), One-shot (O), Transient (T), and variable One-shot or Modal (O-M).

"Figure 8 [p1-11] lists the entire present vocabulary of the APT system and shows whether each word is to be placed in the major or minor section of an APT statement... The words are arranged into classes with appropriate headings such as 'motion instructions' and 'parameter names', and within some of the classes, words are further grouped into sets by means of brackets. Only one word from any set (or if there is no bracket, only one word from the entire class) may appear in any single APT statement, unless a class contains several bracketed sets, in which case one word from each set may appear. There are of course many exclusions from this general rule, e.g., if one word is chosen at random from each of the classes and sets, then a meaningless statement will result. Once the meaning [sic] of the individual words are understood, however, if the words are selected such that they make good English sense, then the resulting APT statement will be acceptable.

"A word which has modal meaning need be written only once in the part program, after which its meaning applies to all succeeding statements until another word from the same set is given... One-shot words apply only to the statement in which they occur and do not modify the meaning of words in other APT statements... One-shot-or-Modal words have either one-shot or modal meaning depending upon what kind of statement they are used in. If they are used in a special statement they have modal meaning whereas if they are used in a definition or instruction statement (i.e., a statement punctuated with a /) their meaning is one-shot. Once a one-shot meaning has been executed, the meaning will revert to the modal setting... Transient words modify only the word immediately preceding or the word immediately following their occurrence. Thus the combination FAR, 3, means the third far intersection; whereas the combination X SMALL, INT OF, means the intersection with the smallest X coordinate. When transient modifiers are bracketed into sets in Figure 8, it means that two words from the same set may not be used to modify the same single word, even though several set uses may appear in a single statement" [pp1-10, 12, 13].

It is, of course, essential to know the meaning of each individual word as well. For example, FROM applies only to a point in space and tells ARELEM the location of the tool following a manual set up procedure by the machine operator. Thus, FROM/LINE would be meaningless and unacceptable. Similarly, INDIR, meaning "in the direction of", works only with points or vectors, not the other geometric quantities.

The first glimmerings of the need for INDIR show in my daily resume for February 25, 1958

[R58225], but the word itself is first referenced in the resume for April 2 and 3 [5842,3], even though it was excluded from the Field Trial language [Ross 1958e]. Along with GO TO, ON, PAST, TAN a surface, INDIR finally solved the start up problem (although ARELEM could not do GO TAN, and probably should not have been asked to). Similarly, allowing TO, ON, PAST, and TAN to be parameters on the right of the slash in a motion instruction, gave complete control over termination of cutting conditions, thereby eliminating the word TERM from the language. To eliminate an illegal jog motion when backing up from a tangent check surface condition, GO BAC became GO BACR or GO BACL to convey the part programmer's intention so that an alarm could be generated in the illegal cases.

Z SURF, which for a long time was restricted to being equivalent to Z PLANE [R58827-29], specified a surface on which all implicitly defined points should lie. I recognized the need for it as early as August 30, 1957 [Ross and Clements 1957], but it never was properly specified, so that the initial preprocessing package [C58324, McWhorter 1958] set the Z coordinate of every point defined equal to zero -- hardly three dimensional.

I invented the "tangent check point", TN CK PT, solution to the tangent check surface problem July 22, 1958 [R58722]. If the step size were selected small enough (limited by MAXDP [R58417-55]) the tangent check surface calculation would be guaranteed to see the check surface if it looked from the current tool position to the tangent check point rather than looking in the surface normal direction. Thus, for example, a single check point interior to a convex part would insure that all surfaces of that part could properly be found. The three LOOK instructions tell whether to use that or the drive surface or part surface normal for tangent check surface calculations. These arose when it was found that the old unreliable surface normal method was in fact preferable for certain classes of parts in which tangent check points would need to be changed repeatedly and yet the normal vector method would work satisfactorily. Hence the part programmer was given this control.

2D CALC and 3D CALC were intended to switch ARELEM calculations from 3 to 2 dimensions and back [Ross and Clements 1957, August 30]. This capability wasn't actually implemented until considerably later.

An interesting phenomenon is that PS IS is all that remains in the 2D APT II system in this initial official language. DS IS and CS IS were later reinstated, but at this point in time, the addition of TO, ON, PAST, and TAN as definition modifiers allowed them to be completely replaced for two-dimensional work, in addition to doing away with word TERM, as has previously been described [Kaynor 1958]. Literally the last word in APT, decided on at the July 2 MIT/UAC meeting, is the word FINI which indicates the very end of each part program so that the APT processing programs can reset to their original condition [Kaynor 1958].

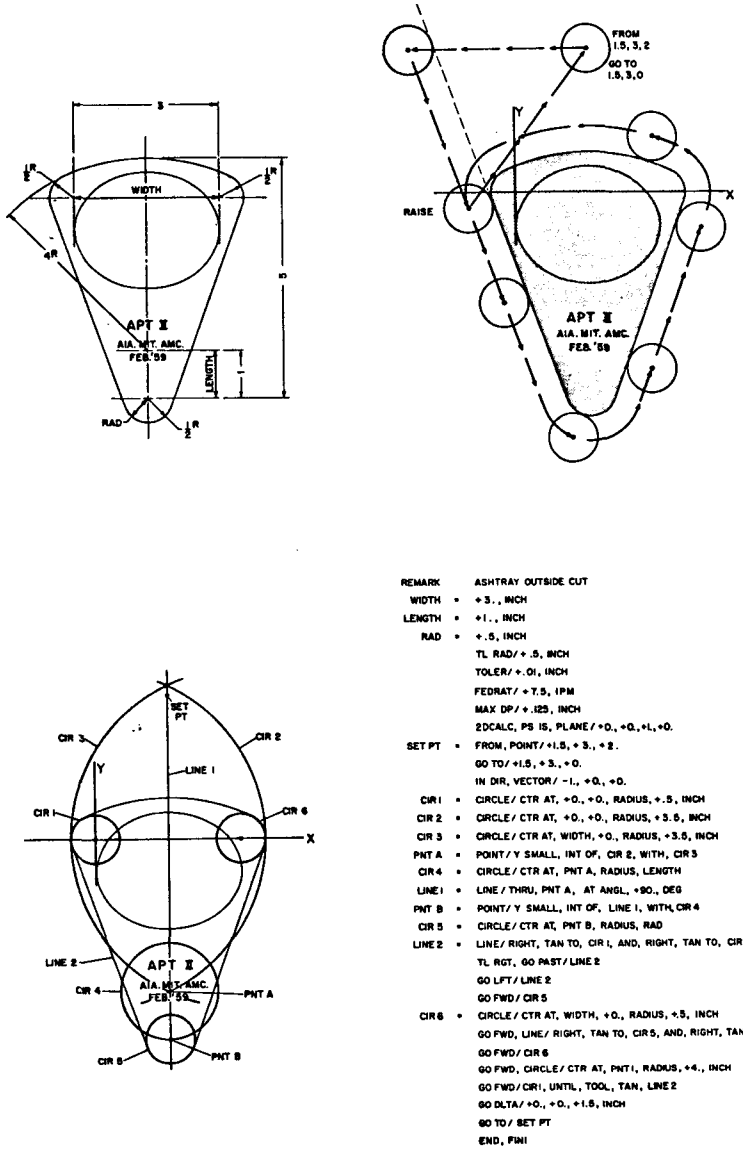
Each geometric name has a corresponding collection of "preprocessing formats" which prescribe the wording and punctuation of the minor section of a statement [Ross 1959a, Ch. 7]. By means of the = sign, a symbolic name can be given to the geometric quantity thus properly defined. By means of synonym statements, symbols may also be assigned for numbers or other symbols. Symbolic numbers may be used wherever numbers are called for and many of the definition formats call for the use of symbolic geometric quantities. If no geometric name is given with FROM, INDIR, GO TO, or GO ON, then point is assumed. Similarly, GO DELTA (the incremental GO TO) has vector as normal case. Ignorables can be written any place to make the reading more English-like, but are completely ignored by the system.

Under parameter names, TOLER was a single mathematical tolerance specification to match the existing ARELEM, even though the need for separately-specifiable inside and outside tolerance had been recognized at MIT since at least January 1957 when the "secant cut" of maximum length allowed by a complete tolerance band was used in APT III [R57017, 8].

Figure 9 shows (the Press Conference ash tray version of) the Geometric Definitions, Tool Path, and Part Program for the first sample part completely worked out in the manual. (The center portion was a three-dimensional "saddle surface" hyperboloid of one sheet, pocketed by decreasing indexed elliptic cylinders, with a slanting gouge for a cigarette at each upper corner.) The example is interesting because it showed the use of the geometric definitions to perform ruler-and-compass constructions to define the complete geometry given very sketchy input requirements. The same outside shape and two other insides of this part were also worked out in the manual. The second outside part program illustrated how the flexibility of the language would allow the part programmer to program a path directly. He could simply start out from the beginning and define quantities as he went along by adding symbols to earlier statements and using those in constructing definitions for later statements, rather than doing all the definitions beforehand as in the first example.

10.2 Phase II Extensions

I don't think there ever was a completely functional APT system corresponding to this first Part Programmer's Manual. Once the Seattle Meeting decided that Phase I would be defined to be "the system that MIT was documenting" I very carefully kept out of the Part Programmer's Manual any ideas that might smack of a possible Phase II [R58725-29]. In fact, it is probably true that it was not until well after the San Diego Project effort in 1961, in which the truly definitive APT system was generated by programmers from many companies working together in the same place for an extended period of time, did APT ever have a set of documentation that adequately matched the system then in use. The whole project was so difficult and drawn out, however, the lack of current documentation seemed like a normal state of affairs to APT participants.




```
REMARK    ASHTRAY OUTSIDE CUT
WIDTH   =  +3., INCH
LENGTH  =  +1., INCH
RAD     =  +.5, INCH
           TL RAD/ +.5, INCH
           TOLER/ +.01, INCH
           FEDRAT/ +7.5, IPM
           MAX DP/ +.125, INCH
           2DCALC, PS IS, PLANE/ +0., +0., +1., +0.
SET PT  =  FROM, POINT/ +1.5, +3., +2.
           GO TO/ +1.5, +3., +0.
           IN DIR, VECTOR/ -1., +0., +0.
CIR 1   =  CIRCLE/ CTR AT, +0., +0., RADIUS, +.5, INCH
CIR 2   =  CIRCLE/ CTR AT, +0., +0., RADIUS, +3.5, INCH
CIR 3   =  CIRCLE/ CTR AT, WIDTH, +0., RADIUS, +3.5, INCH
PNT A   =  POINT/ Y SMALL, INT OF, CIR 2, WITH, CIR 3
CIR 4   =  CIRCLE/ CTR AT, PNT A, RADIUS, LENGTH
LINE 1  =  LINE/ THRU, PNT A, AT ANGL, +90., DEG
PNT B   =  POINT/ Y SMALL, INT OF, LINE 1, WITH, CIR 4
CIR 5   =  CIRCLE/ CTR AT, PNT B, RADIUS, RAD
LINE 2  =  LINE/ RIGHT, TAN TO, CIR 1, AND, RIGHT, TAN TO, CIR 5
           TL RGT, GO PAST/ LINE 2
           GO LFT/ LINE 2
           GO FWD/ CIR 5
CIR 6   =  CIRCLE/ CTR AT, WIDTH, +0., RADIUS, +.5, INCH
           GO FWD, LINE/ RIGHT, TAN TO, CIR 5, AND, RIGHT, TAN TO, CIR 6
           GO FWD/ CIR 6
           GO FWD, CIRCLE/ CTR AT, PNT 1, RADIUS, +4., INCH
           GO FWD/ CIR 1, UNTIL, TOOL, TAN, LINE 2
           GO DLTA/ +0., +0., +1.5, INCH
           GO TO/ SET PT
           END, FINI
```

Figure 9. PART PROGRAM NO. 1
Rocker Arm Cam Outside Cut

A few historic highlights are worth citing even though they don't show in the first APT Part Programmer's Manual because they showed up much later in subsequent development or have not been pinned down as to time, in my previous discussion here.

1) The use of symbolic names for geometric quantities minimizing numerical parameters so that entire families of part shapes can be accommodated by the automatic programming system, comes from my discussions January 29 through 31, 1957 with Arnie Siegel in preparation for the Numerical Control Course at MIT.

2) The various check surface types, including their use as decision surfaces, was seen February 11, 1957 in Whirlwind APT II discussions.

3) February 12, 1957, again in discussions with Siegel regarding the course, is the first place that the terms "preprocessing" and "post-processing" for those major sections of the APT system structure make an appearance.

4) April 29, 1957 I first proposed the tabulated cylinder (which like an aurora borealis passes parallel

lines through an arbitrary chain of points in space) as the general way to handle arbitrary two-dimensional curves in a way that properly fits into the three-dimensional case.

5) On August 15, 1957 NAA raised questions on using filleted end mill cutters to cut pocketing with sloping bottoms as in the AIA Test Part No. 1. They suggest the idea of a "limit surface to control the extent to which any of the other surfaces, part surface, driving surface, check surface apply" [Ross and Clements 1957, August 30, p11]. We actually did incorporate such limit surface capabilities in our MIT ARELEM in 1960, but I believe that this is still a current topic to be accomplished in the current CAM-I Advanced NC Project even today.

6) The "CL tape" cutter location tape standard of APT came out of talks I had at North American with Dale Smith in early October 1957 and was selected by me at Carlberg's urging by "semi democratic authority" in preference to a less-flexible scheme then being proposed by Boeing, on December 3rd, 1957. (It was later refined somewhat.)

7) The first test run of CONTROL and ARELEM together took place sometime in January 14 through 16, 1958 on the IBM 704 at MIT.

8) On March 14, 1958, in his first Coordinator's Report [Smith 1958], Dale Smith presented a proposal for automatic pocketing based upon indexing of drive surfaces. We also had considered indexing in our APT work at MIT, and this type of operation did get incorporated in APT in the APT Macro facility about 1961 [AIA 1961].

9) Under "Loops and subroutines in the part program" Dale also recognized that there is more to indexing, but evidently he was not aware of our suggestion [R57211] that check surfaces be used as actual decision surfaces for part program logic control. I had included labelled instruction statements in my original June 14, 1957 Language Memo [Ross 1957c] for this purpose.

10) We first talked about iterating a general tool shape into the valley formed by multiple check surfaces on April 7, 1958.

11) While writing the Post Coordinator's Report July 25, 1958, I saw (with Pople) that using the terms N, E, S, W would be preferable to the RGT, LFT, FWD, BAC, especially to go with later coordinate transformations, but it never got into APT.

12) On October 17th through 20, 1958, I say, "Talking a little bit more long range with Clare, I pointed out that we could use the connectives AND and OR really with no need for parentheses since we would want to follow any such statement by suitable IF clauses which would automatically determine the meaning. This is somewhat different that FORTRAN since we can, in our case, make the system figure out what we mean by what we say in later statements. In strictly mathematical problems this is not possible.... The problem of macrocoding such as is required in tool shape and director specifications, [also mentioned in the March 14 Dale Smith report] as well as the inclusion of FORTRAN and SAP language in APT programs also should be included in Phase II. The writing of a general normal vector and directed distance program would facilitate the inclusion of arbitrarily defined surfaces."

These not only presage the phrase substitution of AED, but were in fact included in my later recommendations to the APT project which became viable features in subsequent versions. It may be of interest to APT afficianados to know that their consideration was documented this far back in time.

10.3 Macros and Phrase Substitution

One of the most interesting aspects of the extensions of APT beyond the initial Phase I language was the way in which several of the more powerful language and translation features that would later become important also in general-purpose programming languages arose in a completely natural fashion much earlier in APT. I believe that the reason this happened is that because of its subject matter (and perhaps any truly application-oriented language would share this property) APT had to be, from the beginning, what I now call an "object-oriented language". [Ross 1975]; in other words, a language in which the viewpoint and terminology reflected in the vocabulary, syntax, semantics, and even the pragmatics of the language are concerned directly with describing and manipulating specific real objects of the area of discourse. The attitude of the user and of the designer of the language is concerned directly with the subject matter and

not with how that subject matter may be represented in a computer. This means that the language is concerned exclusively with objects, and with properties of or operations on those objects. (This means that the types, in the sense of today's abstract data types [Liskov 1978, Shaw 1978, etc.], are concerned with descriptors and operators of the type.) There is no place in either the thinking or the language for "pointers" or "references", which are strictly representational matters.

In the case of APT, two important features came up early and naturally -- macros and nested definitions (i.e., phrase substitution). The first ideas for macros in APT (which I have previously cited) have to do with making repetitive patterns of holes, (DRILL), or in carrying out the reliable control of feedrate in the drilling-like start-up of a pocketing operation, (PLUNGE). These were considered macro instructions and called "macro-instructions" [Smith 1958; 1, p10] because they involved repetitive patterns of the more elementary GO types of APT instructions. It was then but a short step to see that the same process applied to both rough and finish cuts in part programming, from which the application of the macro pattern-generation concept applied equally well in the motion-instruction domain in general. This in turn brought out the idea that the same concern for safe control of feedrate during a plunge operation should be applied to the same concerns for general part programming, and the idea of packaging stylized patterns of general part-programming-standards macros was a natural consequence.

All of this took place well before the development of the SAP macro facility in 1960 by McIlroy of Bell Labs [McIlroy 1960]. (My resume of February 10 through 29 [sic!], 1960 says "Got a writeup of the new SAP MACRO system just this morning. It appears that these facilities will suit very well our flow diagramming and coding conventions and will make the generation of the compiler or whatever system I ultimately end up with much easier and more powerful. The boys from Bell Labs did a very good piece of work here" [R60210-29]. In fact, our early APT macro thoughts included conditional macros, for that is how "loop-logic" was to be exploited for rough and finish cuts. This was, in fact, the purpose of the "symbol) ..." beginning of the statement types in my original June 14, 1957 Language Memo [Ross 1957c], although it was dropped for a time and did not come back until Dale Smith's Coordinator's Report #4 of August 20, 1958 [Smith 1958, 4] lists "symbolic statement identification" along with "completely variable field card format." But it is not until his "Prognostications" of report #7, dated January 13, 1959 [Smith 1959] that we find among other things "Addition of logical part program jump instructions" and "Addition of macro instructions" as separate headings. I believe it was sometime after that that I finally was able to start getting people to see that from the beginning I had intended that these all be one set of thinking.

The matter of nested definitions or phrase substitution, in which any phrase of a given type can be substituted wherever any other phrase of the same type can occur, may go back as far as my original discussions with Arnie Siegel, especially in the context of the preparation for the Numerical Control Course we gave for AIA at MIT [R57212], but I can't really vouch for that. The earliest APT language, like Siegel's language, only allowed symbols to be given to geometric quantities and then those symbols were written in instructions or other definitions -- but not the definitions themselves. I do know, however, that once we had seen that DEFPRE could be called from INSPRE [R58930-107], (and I spoke of it at the time as "nesting definitions in instructions") the idea of nesting definitions within definitions was a natural step. The problem was that the definition preprocessor control was not up to that task at the time. (It was a simple dispatcher, not a compiler.) Therefore it is not until the September, 1959 Coordinator's Report of Len Austin (who followed Dale Smith as Coordinator) that we find nested definitions as an assigned APT Phase II item [Austin 1959]. I had been pressing for its inclusion ever since writing the portion of the Phase I Part Programmer's Manual in which instruction statements were converted into definition statements after the fact by writing "symbol = ..." at the beginning of an already-written statement in the "path-oriented" second program of the rocker arm can sample part (see Section 10.2).

The important thing to notice about these macro and nested definition language features is that they both became full-blown in concept and purpose, if not in practice, at least by early 1959 and certainly were brewing in 1957. My recollection is that they came from the needs of the application area, rather than from any influences of general-purpose programming thinking (although I can't fully vouch for that). In any case they were very influential on my own formulation of plex and my Algorithmic Theory of Language which, as I have already cited, also had its roots in the same period. [Another interesting aside is that my resume shows that on April 16 through 18, 1959 a symbolic manipulation conference was held at MIT (no formal auspices) at which "I talked for a short time on multimode control as applied to list searches, [Ross 1958d], group control and proposed a modified list structure which seems more appropriate to our design machine application". [R59416-511] which was my first public discussion of my "n-component element" plex ideas (although I did not at that time have the word "plex"). This predates by almost exactly one year the May 20-21, 1960 Philadelphia ACM conference on Symbol Manipulation at which my note published in CACM March 1961 [Ross 1961] was presented. As I have cited here, the start of this thinking was "Milling Machine Conference Number One" in 1956 [Ross 1956c]. Since most people seem to associate 1961 (the CACM late publication date) with the start of my plex thinking, I thought it would be interesting to point out these earlier dates.]

11. PERIOD 9: THE PRESS CONFERENCE (November 1958 through February 1959)

The official launching of APT on the world was, of course, the Press Conference of February 25, 1959 [MIT 1959a], with which I opened this paper. According to my resumes, over a year went into its preparation [R580131], triggered by the significant press coverage of my AAAS paper

reverberating through the month of January 1958. Nothing actually happened, of course, for a long time as the Field Trial effort struggled past the projected May 1st date. On June 9, 1958 I received an invitation to present a paper at the ASME annual meeting in New York City December 1 through 5, 1958, in a special program sponsored by the Aviation and Machine Design Divisions [C5869]. I accepted and proposed the title "A Progress Report on the 2D APT II Joint Effort for Automatic Programming of Numerically Controlled Machine Tools" [C58630]. On August 13, I started writing, and by August 29 I was putting the finishing touches on the ASME paper [R58827-29]. A figure in that paper [Ross 1958g] shows the state of my thinking at that point in time with regard to the Figure 8 language chart of the final Part Programmer's Manual. At that point in time, "miscellaneous" words was a catch-all category. I had not yet decided to treat INDIR as a motion instruction because it set the sense of direction for all the rest. ZSURF was ZPLANE at that point, and direct setting of the drive and check surfaces was included. A few spellings are also different, but the language itself was almost complete when this paper was submitted on August 29, 1958.

By mid November George Wood of MIT public relations department took over [R581113 + 14]. His primary suggestion was that we try to arrange some sort of demonstration along with the release. By January 9 we had decided to make souvenir ashtrays from the part programming example and have sense switch control of scope plots, etc. [R581229-59019]. The Press Conference itself took place February 25, 1959.

> "We had a little over 60 press people present, split almost equally between popular and technical press. The impact of the conference was, I think, beyond anyone's expectations. We evidently made the front page several places and received some radio and TV coverage as well... Lowell Thomas also made a pun on APT, saying that with APT almost anything was apt to happen... One of General Irvine's statements was taken out of context so that the press releases said that he said that APT would make possible a war machine that the Russians wouldn't dare to tackle. Whereas actually what he had said was more of this kind of development could lead to such a situation. We haven't yet begun to get the technical press and magazine coverage which should be more factual than the newspaper coverage... The amount of notoriety that has reflected on me personally is quite unnerving but I seem to be surviving well enough and I hope my professional colleagues will take most of the noise with a grain of salt" [R59219-34].

It was, however, a first class job all around with a much higher than usual communication of valid technical information into the public consciousness. It was, I think, more than mere hoopla, for the added credibility of this widespread attention helped to see the APT Project through the many ups and downs in the later months and years. (The resume for January 29 - February 5, 1959 says "We are going to call this the APT Project rather than the Joint Effort".)

12. PERIOD 10: THE MIT ARELEM EPILOGUE (July 1959 through July 1962)

Although APT was used for production parts from mid-1959 on, in various places, the performance of ARELEM was extremely erratic and unreliable. To wrap up the story and provide a tie to my later development of plex, my Algorithmic Theory of Language (the application of plex to the subject of language), and AED, I will cite briefly the subsequent history.

As we were finishing up the Phase I documentation, in the summer of 1959, the AIA APT Project asked us to look into the ARELEM problem at MIT. Even though we were switching our Air Force sponsorship from APT to the subject of computer-aided design and computer graphics, we carried along an APT collaboration task until July of 1962. During that period, with Clarence G. Feldmann (and part of the time with John F. Walsh) I worked out a completely new approach which became known as the "MIT ARELEM", which still forms the basis for the APT system used widely today [MIT 1961]. We did two complete versions of this ARELEM analysis, the First-Order Analysis being based primarily upon linear approximations, the Second-Order Analysis directly taking into account the curvature approximations to yield much more efficient results. Both analyses gave "guaranteed cut vectors" in the sense that using the actual points of tangency to the surfaces involved, the "deepest point of gouge" was determined with respect to independent inside and outside tolerance bands at both ends of the straight-line cut vector, as well as in the middle of the cut. We also, in our MIT version, had a scheme called "parabolization" which is a generalized scheme for drastically speeding up and guaranteeing the convergence of iterative vector computations. Both versions of the program also incorporated adaptive "learning" to generate excellent starting conditions for the cut vector iterations. The scheme for efficiently selecting the appropriate tool segment for use with a given center of curvature of a surface is interesting because it corresponds almost exactly to the guarded commands of Dijkstra [1975], including several very efficient algorithms for evaluation of "simultaneous Boolean expressions" with a minimum number of steps [MIT 1961, pp 61-66].

Although we had all of these versions of the MIT ARELEM working at MIT at various times, it was a lengthy process and went through several stages of interaction with the on-going AIA APT Project. The very first incomplete version was made mostly workable by programmers from Chance-Vought and was used in AIA APT for a short time even though we were continuing with the developments further at MIT. Feldmann spent many months and many trips working directly with the San Diego APT Project, first helping them to program the First-Order Analysis in FORTRAN, and later incorporating portions of the Second-Order Analysis as well. To our knowledge the AIA APT never did incorporate the complete MIT

Second-Order Analysis with parabolization, and the work had dragged out to such an extent and we were so deeply into the beginnings of AED by 1962, that we never did complete the documentation of the final analyses beyond the quite complete descriptions that appeared in our MIT project Interim Reports. By that time, however, the APT Long Range Program was beginning at Armour Research Foundation (now IITRI) [Dobe 1969] so that the APT Project managed to get along without our further intensive assistance.

The programming and testing of the MIT ARELEM from 1959 through 1961 was the place where we first worked out the ideas and approaches to n-component element plex programming using "reverse index registers!". Each tool segment and each surface had many parameters which were kept together in a block of contiguous storage with different components accessed by an appropriate setting of an index register which pointed to the head of the block and the offset with respect to the index picked up the component. We also used the same method to embed transfer instructions in the n-component elements so that an indexed transfer jump through the element would act as a switch in a program. In this way merely by changing the index register pointer from one tool segment to another or one surface to another, the entire collection of programs constituting ARELEM would radically change both its data and its flow-of-control behavior. Although now commonplace in advanced programming languages, this was a radical innovation in 1959. It was, however, very natural in the context of our ARELEM analysis and the nature of the problem we were solving.

One thing that has always disappointed me is that this entire massive effort had so little direct impact on the other developments in programming language and computer science developments. For the most part the (by now) several hundred computer programming people who have worked on the system have been from industry, not academia. Paper writing, except addressed to the "in-group", has been almost non-existent. The continual evolution of more sophisticated features in systems which were in daily productive use meant that almost never was a complete, elegent package available for concise description and documentation. Finally, the system and problem area itself is so complex and elaborate that it is difficult to isolate a single topic in meaningful fashion for presentation to those not familiar with the complete APT and numerical control culture. For the most part, APT was viewed with respect, but from a distance, by members of the computer science community, even though many of the significant developments of the '70s first were tried out in the '60s in APT.

In any case, I hope that this historical look into the beginnings of APT has helped to redress the balance, somewhat. I believe the hypothesis of my theme -- that although old, the original APT was "modern", because otherwise it could not have been done -- is supported by the facts from which I found it emerging. The one solid conclusion I have reached is that it is much easier to make history than to write about it. I now have added to my APT files endless notes and cross references that may, someday, be useful to real historians. All I can really say is that an awful lot of people did an awful lot of good work in those days. I'm glad I was part of it all.

REFERENCE LIST

AIA. 1957a March 26. Report of the Meeting of the AMEC/SNC Study Group for Manuscript Codes, Computer Programming and Computer Sub-Routines Held on 1 March 1957 in Los Angeles, California. Los Angeles, CA: Aircraft Industries Association memo AMEC-57-34.

——. 1957b April. Report Concerning the Meeting of the AIA/AMEC-SNC Control Data Processing Group Held at AIA, Los Angeles, California on 23-24 April 1957. Los Angeles, CA: Aircraft Industries Association (draft copy; no doc. number).

——. 1957c October 18. Report of the AMEC/Subcommittee for Numerical Control held at AIA, Los Angeles, California, on 3-4 October 1957. Los Angeles, CA: Aircraft Industries Association memo AMEC-57-87.

——. 1958 August 20. Report of AMEC/Numerical Control Panel Meeting held in Seattle on July 16-17-18, 1958. Los Angeles, CA: Aircraft Industries Association memo AMEC-58-44.

——. 1959 October 9. Meeting Report, Proceedings of Computer Programmers Meeting, August 24-26 (Project 358-12.3). Los Angeles, CA: Aerospace Industries Association memo MEC-59-69. (Section VIII, p12, starts MIT ARELEM work.)

——. 1961. APT Documentation (6 Volumes) Washington, DC: Aerospace Industries Association (results of the APT III Central Project at San Diego, CA).

ANSI. 1976. AmericanNational Standard Programming Language PL/1. New York, NY: American National Standards Institute, Inc. Doc. No. ANSI X3.53-1976.

Austin, L. 1959 September. Summary of September 1959 Monthly APT Progress Reports. St. Louis, MO: McDonnell Aircraft Corporation (no number).

Benetar, V. 1957 May 10. Subject: Standard Manuscript Language, Marietta, GA: Lockheed Aircraft Corp. memo to AIA AMEC/Subcommittee for Numerical Control (no number).

Boeing. 1957a February 27. Part Programming Language, Numerical Control Program - Tentative. Seattle, WA: Boeing Airplane Co. Numerical Control Mathematical Programming memo, 1 page.

———. 1957b February 28. Numerical Control Library Routines A.M.C. Skin Mills - Preliminary Outline. Seattle, WA: Boeing Airplane Co. memo, 16 pages.

Braid, I. C. 1975 April. The Synthesis of Solids Bounded By Many Faces, Communications of the ACM 18(4):209-216.

Bromfield, G. 1956 January 12. Numerical Control for Machining Warped Surfaces. Cambridge, MA: MIT Servo Lab Rpt. No. 6873-ER-14.

Carlberg, E. F. 1957 March 27. Letter to D. T. Ross with Attachments A-F.

Clements, D. F. 1957 June 21. Coordinator's Report for Period May 20 - June 20. Cambridge, MA: MIT Servo Lab memo 2D APT II-6.

———. 1958 March 4 letter to O. D. Smith.

Coons, S. A. and Mann, R. W. 1960 October. Computer-Aided Design Related to the Engineering Design Process. Cambridge, MA: MIT Servo Lab Rpt. No. 8436-TM-5. 13 pages. DDC No. AD252061.

Dijkstra, E. W. 1975. Guarded Commands, Nondeterminacy and Formal Derivation of Programs. Communications of the ACM 18(8): pages 453-457.

Dobe, J. W. 1969 April. The APT Long Range Program: Progress to Date; Plans for the Future. Glen View, IL: Numerical Control Society Sixth Annual Meeting and Technical Conference Proceedings.

Everett, R. R. 1951. The Whirlwind I Computer. New York, NY: AFIPS, Proceedings of the 1951 EJCC: page 70.

Gregory, R. H. and Atwater, T. V. Jr. 1956 March 1. Economic Studies of Work Performed on a Numerically Controlled Milling Machine. Cambridge, MA: MIT Servo Lab Rpt. No. 6873-ER-18. See also 1957 Journal of Engineering 8(6): 337-352.

Hori, S. 1972 July. The Structure of Functions and its Application to CAM Planning. Glen View, IL: Numerical Control Society NC Scene July 1972: pages 2-5.

IBM. 1963 January. ADAPT, A System for the Automatic Programming of Numerically Controlled Machine Tools on Small Computers. San Jose, CA: Final Tech. Eng. Rpt. (Air Force Contract AF33(600)-43365).

Kaynor, K. 1958 July 8. Memo to R. Nutt. Subject: Conclusions reached at MIT on July 2, 1958.

Kinney, G. E. 1957 June 10. Letter to G. W. Jacob. Copy received by D. T. Ross 1957 June 18.

Liskov, B., Snyder, A., Atkinson, R., and Schaffert, C. 1978 August. Abstraction Mechanisms in CLU. Communications of ACM 20(8): 564-576.

McIlroy, M. D. 1960 April. Macro Instruction Extensions of Compiler Languages. Communications of the ACM 3(4): 214-220.

McWhorter, B. J. 1958 April 10. Letter to D. T. Ross, with enclosures. Contains excellent description of INTRAN-DEFPRE processing method.

MIT Servo Lab. 1952 July 30. Final Report on Construction and Initial Operation of a Numerically Controlled Milling Machine. Cambridge, MA: Report No. 6873-FR-1. Reprinted in Appendix B of [Ward 1960].

———. 1956 March 15. Design, Development and Evaluation of a Numerically Controlled Milling Machine. Final Report. Cambridge, MA: Report No. 6873-FR-2. Reprinted in Appendix A of [Ward 1960].

———. 1957a January 1 through March 31. Automatic Programming for Numerically Controlled Machine Tools. Cambridge, MA: Rpt. No. 6873-IR-3.

———. 1957b February 18. Programming for Numerically Controlled Machine Tools. Cambridge, MA. Special course brochure printed by MIT Summer Session Office.

———. 1957c March 23-April 3. Course Outline and Workbook for the Special Course on Programming for Numerically Controlled Machine Tools. Cambridge, MA: (no report number).

———. 1958a January 1 to June 30. Automatic Programming of Numerically Controlled Machine Tools. Cambridge, MA: Rpt. No. 6873 IR-6 and 7, ASTIA No. AD-156060.

———. 1958b April. Research in Defense Techniques for Airborne Weapons, 1957 Annual Report; Vol. 2. Cambridge, MA: Servo Lab Rpt. No. 7668-R-5(2).

———. 1959a February 25. APT Press Conference. Cambridge, MA: Appendix C of [Ward 1960].

———. 1959b. APT System Documentation.

Ross, D. T. 1959b June. Vol. I, General Description of the APT System, 85 pages.

———. 1959a May. Vol. II, APT Part Programmer's Manual, 130 pages.

XXXXX. Vol. III, APT Calculation Methods (not published).

MIT and AIA participating company staffs, 1959 May. Vol. IV, A Description of the APT Computer Programs, 162 pages.

Feldmann, C. G. 1959a May. Vol. V, Operator's and Troubleshooter's Manual, 27 pages.

———. 1959b May. Vol. VI, Modification and Change Procedures, 54 pages.

McAvinn, D. 1959 December. Vol. VII, Group Control for Automatic Manipulation of Computer Programs which Exceed Core Memory, 59 pages.

———. 1961 January. Investigations in Computer-Aided Design (December 1, 1959 to May 30, 1960). Cambridge, MA: Interim Engineering Rpt. No. 8436-IR-1.

Pease, W. 1952 September. An Automatic Machine Tool. Scientific American. 187(3): 101-115.

Perlis, A. J. 1958 March 27. Letter to D. T. Ross rejecting [Ross 1957f] for publication in CACM.

Ross, D. T. 1956a February 7-9. Gestalt Programming: A New Concept in Automatic Programming. New York, NY: AFIPS, Proceedings of the 1956 WJCC, pages 5-9.

———. 1956b through 1963. Daily Resumes (Unpublished). Lexington, MA: to be placed in MIT Archives, 833 pages.

———. 1956c November 30. Machine Tool Programming Conference No. 1 (Unpublished memo draft), Cambridge, MA.

———. 1957a March 29. Design of Special Language for Machine-Tool Programming. Cambridge, MA: published in [MIT 1957c], pages 3/29.5-9. (Reproduced here in Section 4, in full.)

———. 1957b May 1. Preparations for Joint Programming of AIA APT II System. Cambridge, MA: MIT Servo Lab. Rpt. No. 6873-TM-2 (Distributed to AIA/AMEC/Subcommittee for Numerical Control)

———. 1957c June 14. A Proposed Basic Language for the 2D APT II. Cambridge, MA: MIT Servo Lab. memo 2D APT II-2, 6 pages (Reproduced here in Section 6, in full.)

———. 1957d October 23-25. Some Recent Developments in Automatic Programming of Numerically Controlled Machine Tools. Presented at Third Annual Contour Machining Conference (no Proceedings). Published in [Ross 1958a].

———. 1957e November 27 letter to B. J. McWhorter.

———. 1957f December 28. Development of a Research Effort in the Automatic Programming of Numerically Controlled Machine Tools. Presented at Association for Computing Machinery Session of the Indianapolis meeting of the American Association for the Advancement of Science (no Proceedings). Published in [Ross 1958a].

———. 1958a January 7. Papers on Automatic Programming for Numerically Controlled Machine Tools. Cambridge, MA: MIT Servo Lab. Rpt. No. 6873-TM-3.

———. 1958b April. The SLURP System for Experimental Programming; Section III-E in [MIT 1958b].

———. 1958c April. A Philosophy of Problem Solving; Section III-D in [MIT 1958].

———. 1958d April. A Multi-Mode Control Element; Section III-C in [MIT 1958].

———. 1958e April 29. 2D-APT II Post Programmer's Manual (Field Trial Version). Cambridge, MA: MIT Servo Lab. memo 2D APT II-16.

———. 1958f July 25. Post Coordinators [sic] Report re Phase I System. Cambridge, MA: MIT Servo Lab. memo 2D APT II-19.

———. 1958g November 30. A Progress Report on the 2D-APT-II Joint Effort for Automatic Programming of Numerically Controlled Machine Tools. New York, NY: ASME Paper No. 58-A-236 at ASME Annual Meeting. Published in condensed form, two parts, 1959 May, Mechanical Engineering, 81(5): 59-60 and 70. Also published as Chapter II in [MIT 1958a].

———. 1959a May. APT Part Programmer's Manual. See Vol. II of [MIT 1959b].

———. 1960 September. Computer-Aided Design: A Statement of Objectives. Cambridge, MA: MIT Servo Lab. Rpt. No. 8436-TM-4, DDC No. AD252060, 22 pages.

———. 1961 March. A Generalized Technique for Symbol Manipulation and Numerical Calculation. Communications of the ACM 4(3): 147-150.

———. 1962 November. An Algorithmic Theory of Language. Cambridge, MA: MIT Servo Lab. Rpt. No. ESL-TM-156, DDC No. AD296998, 68 pages.

———. 1964 September. AEDJR: An Experimental Language Processor. Cambridge, MA: MIT Servo Lab. Rpt. No. ESL-TM-211, DDC No. 453881, 53 pages.

———. 1975 December. Plex 1: Sameness and the Need for Rigor and Plex 2: Sameness and Type, with "are: pres. pl. of BE" [1976 April]. Waltham, MA: SofTech, Inc. Rpt. Nos. 9031-1.1, 2.0, and 10. (Abstracted in [Ross 1976]).

———. 1976 March. Toward Foundations for the Understanding of Type. SIGPLAN Notices 8(2), Vol. II, Proceedings of Conference on Data: Abstraction, Definition and Structure: pages 63-65. (Abstracted from [Ross 1975].)

———. 1977a January. Structured Analysis (SA): A Language for Communicating Ideas. IEEE Transactions on Software Engineering, 3(1): 16-34.

———. 1977b October. Comments on APT Items in D. T. Ross Daily Resumes (unpublished) Lexington, MA.

———— and Clements, D. F. 1957 and 1958. Coordinator's Report(s), Cambridge, MA: MIT Servo Lab. memos

2D APT II-9 for Period August 1 - August 30

2D APT II-10 for Period September 1 - September 27

2D APT II-11 for Period September 28 - October 21

2D APT II-12 for Period October 22 - February 4, 1958

2D APT II-13 for Period February 5 - March 13

2D APT II-14 for Period March 14 - April 4

2D APT II-15 for Period April 5 - April 15

2D APT II-17 for Period April 15 - May 1

———— and McAvinn. 1958 December. Data Reduction for Pre-B-58 Tests of the XMD-7 Fire-Control System, Vol. 3 Evaluation of Fire-Control System Accuracy. Cambridge, MA: MIT Servo Lab. Rpt. No. 7886-R-3, ASTIA AD 207 353.

———— and Pople, H. E. Jr. 1956 June 26 through December 31. Automatic Programming of Numerically Controlled Machine Tools. Cambridge, MA: MIT Servo Lab. Rpt. Nos. 6873-IR-1 and 6873-IR-2.

————, Rodriguez, J. E., and Feldmann, C. G. (Ed.). 1970 January. AED-0 Programmer's Guide. Cambridge, MA: MIT Servo Lab Rpt. No. ESL-R-406 published by SofTech, Inc., Waltham, MA.

Runyon, J. H. 1953 December 1. Whirlwind I Routines for Computations for the MIT Numerically Controlled Milling Machine, Cambridge, MA: MIT Servo Lab. Rpt. No. 6873-ER-8.

Shaw, M., Wulf, W. A., London, R. L. 1977 March. Abstraction and Verification in ALPHARD: Defining and Specifying Iteration and Generators. Communications of the ACM 20(8): 553-564.

Siegel, A. 1956a March 1. Information Processing Routine for Numerical Control. Cambridge, MA: MIT Servo Lab. Rpt. No. 6873-ER-16.

————. 1956b October. Automatic Programming of Numerically Controlled Machine Tools. Control Engineering, 3(10): 65-70.

Smith, O. D. 1958 and 1959. AIA Coordinator's Report(s). Los Angeles, CA: Aircraft Industries Association memos

1. AMEC-58-17; 1958 April 4; for Period through 1958 March 14
2. AMEC-58-45; 1958 August 25; for Period through 1958 August 7
3. AMEC-58-47; 1958 August 27; Definition Preprocessing Memo
4. AMEC-58-47; 1958 August 27; for Period through 1958 August 20
5. AMEC-58-55; 1958 October 8; for Period through 1958 September 9
6. AMEC-58-62; 1958 October 30; for Period through 1958 October 22
7. AMEC-59-5; 1959 January 20; for Period through 1959 January 13
8. AMEC-59-11; 1959 February 24; Work Assignments 1959 February 18

———— and Corley, C. F. 1958 February 10. APT-II Arithmetic Program. Los Angeles, CA: North American Aviation, Inc. SHARE-type writeup submission to 2D APT II Field Trial.

Swift, C. J. 1957 October 24 letter to E. F. Carlberg.

Voelcker, H. B. and Requicha, A. A. G. 1977 December. Geometric Modeling of Mechanical Parts and Processes. IEEE Computer 10(2): 48-57.

Ward, J. E. 1960 January 15. Automatic Programming of Numerically Controlled Machine Tools. Final Report. Cambridge, MA: MIT Servo Lab. Rpt. No. 6873-FR-3.

————. 1968. Numerical Control of Machine Tools. New York, NY: McGraw Hill Yearbook of Science and Technology: pages 58-65.

Wirth, N. and Hoare, C. A. R. 1966 June. A Contribution to the Development of ALGOL. Communications of the ACM 9(6): 413-431.